SPACELAB SYSTEM ANALYSIS:

THE MODIFIED FREE ACCESS PROTOCOL:

AN ACCESS PROTOCOL FOR COMMUNICATION SYSTEMS WITH PERIODIC AND POISSON TRAFFIC

CONTRACT:

NAS8-36717

FINAL REPORT

10/31/89

FRANK INGELS

JOHN OWENS

PRINCIPLE INVESTIGATORS

STEVEN DANIEL

TECHNICAL INVESTIGATOR

ELECTRICAL ENGINEERING DEPT.

MISSISSIPPI STATE UNIVERSITY

601-325-3912

SPACELAB SYSTEM ANALYSIS:

THE MODIFIED FREE ACCESS PROTOCOL:

AN ACCESS PROTOCOL FOR COMMUNICATION SYSTEMS WITH PERIODIC AND POISSON TRAFFIC

CONTRACT:

NAS8-36717

FINAL REPORT

10/31/89

FRANK INGELS

JOHN OWENS

PRINCIPLE INVESTIGATORS

STEVEN DANIEL

TECHNICAL INVESTIGATOR

ELECTRICAL ENGINEERING DEPT.

MISSISSIPPI STATE UNIVERSITY

601-325-3912

NASA

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle

SPACELAB SYSTEM ANALYSIS:
THE MODIFIED FREE ACCESS PROTOCOL:
AN ACCESS PROTOCOL FOR COMMUNICATION SYSTEMS
WITH PERIODIC AND POISSON TRAFFIC

5. Report Date
October 1989

6. Performing Organization Code

7. Author(s)

F. M. Ingels
J. K. Owens
S. P. Daniel

8. Performing Organization Report No.
MSU-EE-FIN-20A-89

10. Work Unit No.

9. Performing Organization Name and Address

DEPARTMENT OF ELECTRICAL ENGINEERING
MISSISSIPPI STATE UNIVERSITY
MISSISSIPPI STATE, MS 39762

11. Contract or Grant No.
NAS8-36717

13. Type of Report and Period Covered
FINAL REPORT Option 3
Oct. 88-Oct. 89

12. Sponsoring Agency Name and Address

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
WASHINGTON, D.C. 20546-0001

14. Sponsoring Agency Code

15. Supplementary Notes

PREPARED BY MISSISSIPPI STATE UNIVERSITY FOR SHERMAN JOBE

16. Abstract

The protocol definition and terminal hardware for the modified free access protocol, a communications protocol similar to Ethernet, are developed. A MFA protocol simulator and a CSMA/CD math model are also developed. The protocol is tailored to communication systems where the total traffic may be divided into scheduled traffic and Poisson traffic. The scheduled traffic should occur on a periodic basis but may occur after a given event such as a request for data from a large number of stations. The Poisson traffic will include alarms and other random traffic. The purpose of the protocol is to guarantee that scheduled packets will be delivered without collision. This is required in many control and data collection systems.

The protocol uses standard Ethernet hardware and software requiring minimum modifications to an existing system. The modification to the protocol only affects the Ethernet transmission privileges and does not effect the Ethernet receiver.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
LAN - Local Area Network FDDI - Fiber Distributed Data Interface ETHERNET	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
UNCLASSIFIED	UNCLASSIFIED	248	

NASA FORM 1626 OCT 86

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

The Modified Free Access Protocol: An Access Protocol for Communication Systems with Periodic and Poisson Traffic

INTRODUCTION

Present high speed communications protocols of the 1 to 50 megabit/second range such as Ethernet, HYPERchannel, ProNET-10 and MIL-STD-1553B are useful for many communications applications[IEEE83/ NETW82/ PROT86/ MILS87]. The Ethernet and MIL-STD-1553B protocols have readily available commercial integrated circuits to serve as channel interfaces which makes the use of these protocols very attractive. The Ethernet , HYPER-Channel and ProNET-10 protocols are often used for computer system communications between workstations and servers as well as between host computers. The MIL-STD-1553B command/response protocol is useful for communications systems with a central controller such as flight systems or centralized data collection systems. These protocols represent the four basic access types: contention, token passing, command/response and hybrid.

Many studies have been performed to determine the suitability of these protocols in various communications systems. The basic trade-off is between average access time and guarantee of access time and data delivery. Contention protocols allow a station to transmit a packet whenever the channel is perceived as idle. This allows minimum delay under light loads but also allows the possibility of collisions to exist. Token passing protocols guarantee each station sole access to the channel for a specified time during each cycle at the cost of requiring stations with traffic to wait as the token is passed through stations that do not have traffic. These two access types can be combined to form a hybrid protocol. An example of a typical hybrid proto-

col is HYPERchannel. In this protocol a period of prioritized access is followed by a period of CSMA/ACK free access.

Another hybrid protocol of note is L-expressnet which was developed as a communications subnetwork for the C-NET project[BORG85]. L-expressnet is interesting in that it is a virtual token passing protocol implemented with standard Ethernet chips and counter circuits controlling the carrier sense line. In addition to these protocols the development of several hybrid protocols that offer improved access for particular applications has been presented in the literature[SPIE86/ TOKO77/ NUTE84]. None of these protocols, however, provide adequately for a communication system that has large periods of Poisson traffic and a small portion of scheduled traffic. A communication system with this type of traffic could be served by a communications protocol that provides a period of scheduled access followed by a period of free access.

The purpose of this research is to attempt to apply LAN techniques to situations that are usually considered incompatible. The particular application of interest is communication systems that have a majority of Poisson distributed communications but some scheduled heavy burst traffic as may be found in data collection systems and some real time applications. Emphasis will be placed on simple modifications of existing hardware. The objective of this research will be to minimally modify the Ethernet standard protocol to service these applications.

1 NETWORK ARCHITECTURES

In designing a data communication network to connect many data sources to many hosts, the architecture of the interconnection network is chosen based on the priorities of the data system. The priorities usually considered are: system reliability, ease of change in system configuration, data latency, and ability to broadcast from one station to all others. Sometimes the physical network chosen will prevent a desired implementation. As an example, fiber optics are not well-suited to multitap bus structures due to coupling losses incurred at each tap. Rather, fiber optics are better suited to ring or star(cluster) architectures. The general architectures usually considered are the star(cluster), bus and ring. Figure 1.1 illustrates these basic configurations. These architectures reside in layer 1, the physical connection, of the ISO seven layer data communication model as shown in Figure 1.2. Several committees, whose purposes embrace the establishment of standards for communications, including LAN's, have been appointed. These include the following:

NATIONALLY:

ANSI - American National Standards Institute Committee X3

EIA - Electronics Industry Association

NBS - National Bureau of Standards

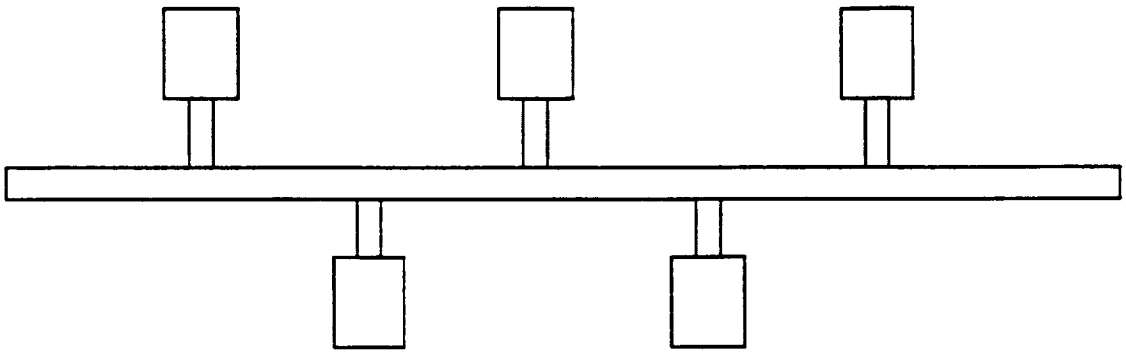
IEEE - Local Network Committee (802)

FTS - Federal Telecommunications Standard 1003

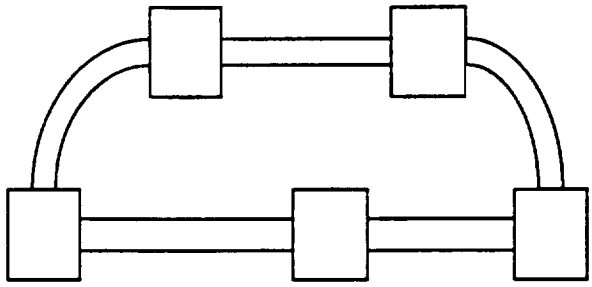
INTERNATIONALLY:

CCIT/ITU - The Consultative Committee on International Telephony and Telegraphy of the International Telecommunications Union, Study Group VII

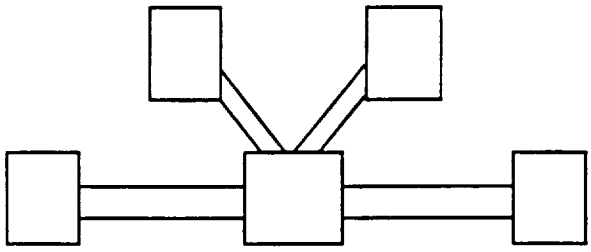
ISO - International Standards Organization



a) BUS



b) RING



c) STAR(CLUSTER)

FIGURE 1.1 Network Architectures

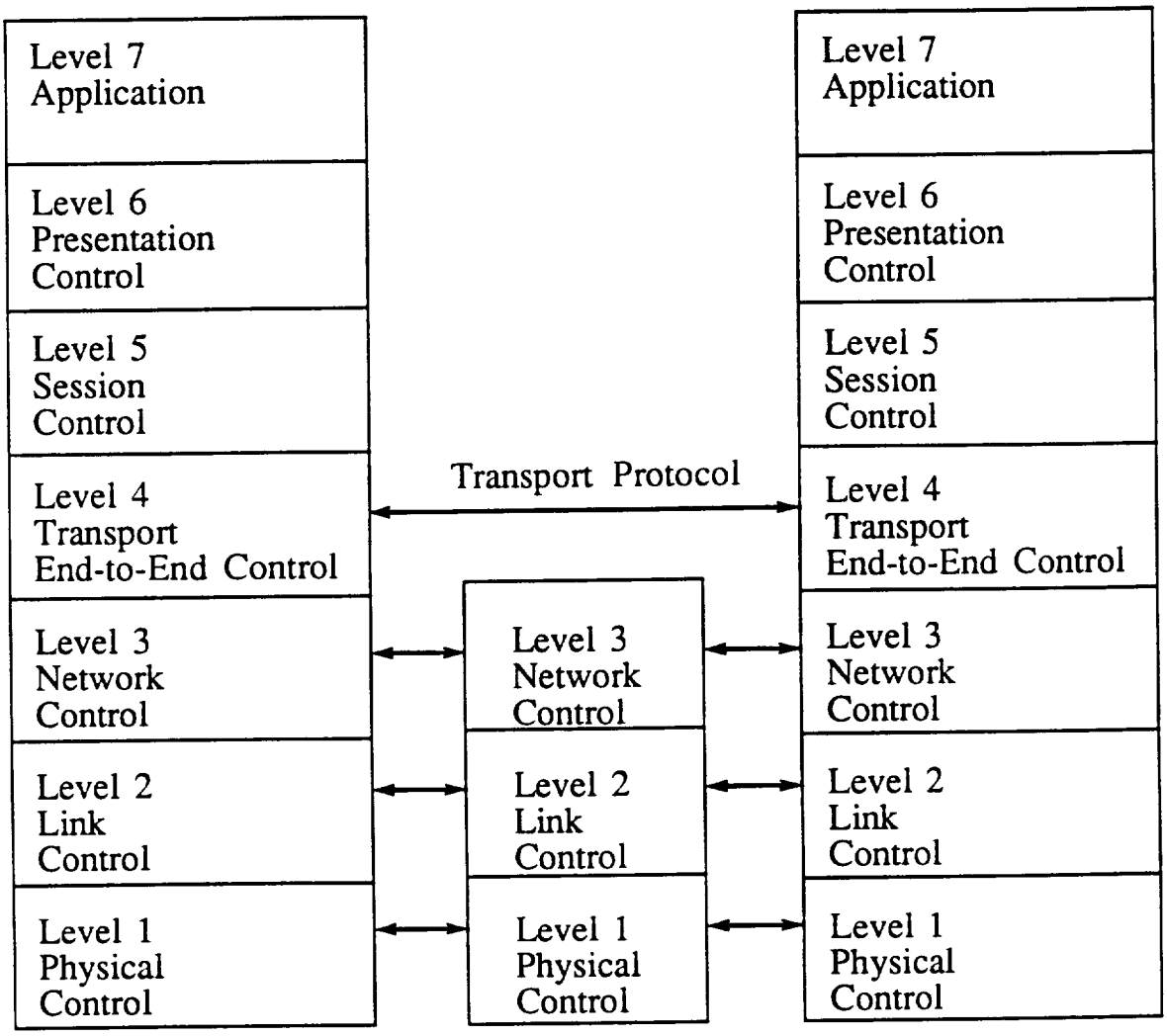


FIGURE 1.2 OSI Seven Layer Communications Model

ECMA - European Computer Manufacturing Association

A seven layer reference model for distributed systems has been developed by ANSI and ISO. It is described as:

Level 1 - Physical Control - the actual means of bit transmission across a physical medium (For example, Manchester Coding via a Coaxial Cable).

Level 2 - Link Control - defines the method for logical sequences of messages to be reliably exchanged across a single physical data link (For example: addressing, packet synchronization, etc.).

Level 3 - Network Control - provides the logical channels capable of routing the message from source to destination via intermediate paths over several data links on a "best effort" basis (For example: addressing, message synchronization from one station through one physical data link (say fiber optic) to a second physical terminal data link on a second media (say coaxial cable) to a destination with two address codes required).

Level 4 - Transport End-to-End Control - provides links for many users across the complete network topology (Error protocol, etc.).

Level 5 - Session Control - supports Level 4 by segmenting and blocking the data required by Level 4 (For example, initiating check point recovery in the event of network failure, etc.).

Level 6 - Presentation Control - provides the formatting of the data being transmitted (Includes encrypting/decrypting, commanding/expanding, transforming data formats, etc.).

Level 7 - Application - functions to be performed by the terminal. Provides for terminal to terminal protocol (Line controller to Host computer data transfer). These

layers are usually accomplished through the mechanism of appending headers to the data that is routed from (or to) the terminal device.

There are several network topologies which are applicable to general data communication systems. These are the RING, BUS, and STAR topologies which are discussed below and illustrated previously in Figure 1.1. Various other topologies (cube and fully connected) are possible, but not implemented. They are either suitable to computer buses or to long haul networks rather than LAN's, or they have properties peculiar to requirements for circuit switching or message switching rather than packet switching.

1.1 RING

Some comments about the ring structure illustrated in Figure 1.1b are in order. If the ring is broken it becomes cumbersome, often impossible, for all stations to talk to each other since each station must receive, regenerate and transmit the data to be passed to the next station. The protocol used is the main determinant of system connectivity after the ring breaks. If token passing is used, there must be lost token finding logic and default station logic added to each terminal so that the ring becomes adaptive in nature. This is not only costly in dollars and power, but it reduces reliability by adding circuitry which can fail. If command response is utilized, then, in effect, the ring is a bus and must function as such. However, each station must still receive the data, interrogate it for address and pass it on. Thus, the ring necessitates a through connection rather than a parallel tap as in a true bus architecture. This allows single point failures in the receiver/transmitter which contribute directly to increased bit error rate and lowered reliability. Since each node must regeneratively repeat the input, so as to retransmit to the next node, as well as pro-

cess the control of the network during its time as network host, each node must be highly reliable. In remote systems which are unattended, this requires hardware redundancy since failure of a node could be catastrophic. Changes to the network, such as adding a terminal, must also be considered, since this results in disabling the ring temporarily and also increases the time for a transmission to circulate around the ring. Although repeaters with bypass switches may be used at each node to help alleviate failure at a node, if the medium used to connect to each node (a cable or a fiber optic cable, for instance), fails, this presents a serious problem. Dual connecting media must be provided if reliability is a strong priority.

1.2 BUS

Bus topology is one of the most reliable of the three topologies considered. There are potential problems, however, even with the bus topology. If the connecting media link is lost, there will be failure of one part of the bus to communicate with the other part. However, on the remaining bus structures, each terminal connected to one or the other pieces of the bus, may still operate. A terminal failure, except for a short circuit failure of the node at the bus, will not present any particular problems to the remaining bus structure since a node is not required to regeneratively produce the incoming data, as in a ring structure. The bus is a passive medium, with each node listening and not playing a part in the communication of every message. This suggests that a network constructed with a bus topology is inherently more reliable than one constructed with a ring topology. Furthermore, this also allows a system configuration change to be rather easy so long as sufficient addressing is available in the protocol. With proper design, the events of catastrophic disruption such as a lightning strike or an errant cross connection to the power lines can be made to affect only the

node involved, be it ring or bus topology. With care, both topologies can be made reliable; but economic considerations favor a choice of the bus system due to its simpler software problems.

1.3 STAR (CLUSTER)

A star or cluster architecture is often inefficient in the amount of cable utilized and the requirement for adding additional input/output (I/O) ports to the central host. The star architecture is, however, tolerant of interconnect problems. If an interconnect cable (which may be coaxial or fiber optic or microwave or any suitable media) breaks, the loss is only that one station connected to the host by that interconnect. Thus, if the system is tolerant to loss of a few of the connected stations, then the interconnects, the transceiver hardware and the necessity of switch over logic is alleviated with the reduced cost and power and, to the extent of lower chip count, a small gain in reliability. The protocol used for accessing will dictate a need for asynchronous or synchronous timing. If synchronous timing is desired, it is necessary to provide the clock to each station from the host. System configuration is easily changed by addition of an I/O port at the host station and modification of software to include the new addressing. It is necessary to plan ahead in choice of host expandability, both in I/O port expandability, memory or buffer storage and power capability. Sometimes host restrictions preclude additional system I/O ports; hence, additional stations, and in this respect, a bus topology has an advantage.

2 NETWORK PROTOCOLS

Network protocols are actually logical rule sets for accomplishing data communications over a given network regardless of the system architecture. In this regard, main communication protocols are considered part of layer 2 (and sometimes extend to layer 3) of the ISO seven layer data communication model. One of the basic protocols is the free access on demand protocol (or contention protocol) based in part on the ALOHA system which was first utilized for computer networking among the Hawaiian Islands[ABRA70]. These protocols are usually modified versions of carrier sense multiple access (CSMA) utilizing collision detection to aid in increasing data throughput. Ethernet uses one such protocol. Another basic protocol is the command/response technique, which is the basis for MIL-STD-1553B[MILS87]. This protocol has one smart control host, and all other stations must respond only when requested by the control host. Token passing is a third basic type of protocol and is similar to the command/response protocol in that the station holding the token is the temporary control station. However, the station token holding time is limited. Generally this time limit is set by the station being assigned to one of three priority levels. The token is passed to the next station after the current token holding station's time expires. Naturally, all stations must contain software and hardware to regenerate lost tokens, stop token ''hogging'' and to compensate for stations that die and cannot pass a token. Protocols such as SDLC and asynchronous protocols are commonly known and can be designed specifically for any system[INTE88]. There are many media access methods. Some of these methods have been built and tested, some are available as commercial systems, and some have been investigated only by simulation and analytical means.

2.1 CSMA-CD

Carrier-Sense Multiple Access combined with Collision Detection(CSMA-CD) represents a very attractive access scheme for a bus system. Under this protocol, every station ready to transmit a packet must listen to the bus to find out whether any transmission is in progress. If there is a transmission in progress, the station defers its transmission until the end of the current transmission. In spite of carrier sensing, packet collisions cannot be completely avoided because of the non-zero propagation delay of the bus. Upon detection of a collision, transmission is aborted, and the station reschedules its packet by determining a random retransmission interval. To avoid accumulation of retransmissions (in other words, to achieve stability) the retransmission interval is adaptively adjusted to the actual traffic load. For example, ETHERNET uses the so-called Binary-Exponential-Backoff algorithm, which means that the average of the retransmission interval is doubled every time a transmission attempt for a certain packet ends in collision[DEC80/IEEE82]. Of course, other control policies are also possible. The CSMA-CD behaves ideally as long as the ratio of propagation delay to mean packet transmission time is sufficiently low. If for reasonable traffic loads, this ratio exceeds 2% to 5% (as a rule of thumb), the increasing collision frequency causes significant performance degradation, i.e., increasing transfer delay and decreasing throughput. For example, in a 10 Mbit/second bus with 2 km cable length, the propagation delay corresponds to approximately 100 bit times. In this case, severe performance degradation is experienced if the mean packet-length is 2 Kilobits or less. Data latency is a point of concern with pure contention systems because there is no guarantee of packet delivery if the system becomes very loaded. Simulation results indicate steeply rising packet delay

times when the system is loaded above 85% throughput[OKAD84]. Furthermore, the throughput has been shown to be a decreasing function (nonstable system) when the old traffic to be retransmitted plus the new traffic to be transmitted (Offered Channel traffic) exceeds 100%.

2.2 TOKEN RING

In a token ring, access to the transmission channel is controlled by passing a permission token around the ring. When the system is initialized, a designated station generates a free token which is passed around the ring until a station ready to transmit holds the token and sends its packet onto the ring. At the end of its allowed transmission time, a sending station passes the access permission to the next station by generating a new free token. This implies that, depending on packet-length and ring round-trip delay, multiple tokens can concurrently exist on the ring. Only one of them, however, can be in the free state. From a reliability and recovery point of view, it may be desirable never to have more than one token on the ring at a time. This can be achieved in two different ways: (1) The sender of a packet does not issue a new free token before it has received its own busy token, (2) The sender of a packet issues a new free token when it has completely erased the packet. These modes of operation differ in their performance characteristics which can be determined based on the analysis of the mean queueing delay for cyclical single-serve queues with switch-over delay. Simulation investigations have shown that the token ring performs well over the whole range of parameters of interest in the local area network[OKAD84]. A necessary condition to achieve this behavior is that the latency per station is kept as low as possible, especially if a large number of stations are attached or short packets are used. If the ring latency is low, no significant benefits can be drawn from

allowing multiple concurrent tokens on the ring. However, for systems with high latency, multiple tokens appear necessary to achieve good performance.

2.3 COMMAND-RESPONSE

MIL-STD-1553B is a document that concerns a command-response local area network[MILS87]. This system is bus oriented in topology and utilizes a central node controller to direct the traffic flow on the bus from terminal to terminal. The network is implemented in a twisted-pair-shielded coaxial cable medium at the present, however, there have been successful fiberoptic implementations. The system is limited in the number of terminals that may be placed on a single bus. This limitation is due to the 5-bit address code in the data link protocol. Hierarchical bussing may be accomplished, but response time becomes a major factor and every addition of a terminal requires a major reprogramming of the central controller. A guaranteed delivery of messages coupled with deterministic data latency timing are distinct advantages of this type of media access method. The protocol is suitable for high duty cycle terminals rather than for bursty or low duty cycle terminals. The bus bandwidth is capable of handling a one megabit data rate; but the protocol imposes some overhead on data transmissions. A more detailed description of this protocol is given in the next section.

3 COMMON HIGH-SPEED PROTOCOLS

In order to determine what protocol is appropriate to service the given communications system, the available protocols should be studied.

In this section, four protocol/architectures for which hardware is commonly available are presented in more detail. The four protocols are Ethernet/Bus, MIL-STD-1553B Command Response/Bus, ProNET-10/Ring, and HYPERchannel/Bus. The protocols' basic characteristics are presented along with the basic hardware required for a standard station.

Emphasis will be placed on the Ethernet protocol because of its readily available hardware. MIL-STD-1553B will be studied in detail because it is a command/response protocol applicable to real time systems. The virtual token passing protocols HYPERchannel and L-expressnet will be studied to determine if virtual token passing should be part of the Modified Free Access Protocol. The ProNET token ring is studied for completeness.

3.1 ETHERNET/BUS

Figure 3.1 illustrates a possible technique which could be used to implement the network interconnection. The use of a contention protocol requires the provision of software, buffer storage and the Ethernet hardware as illustrated.

Ethernet is a burst transmission protocol well suited to spasmodic transmission of data from each station. For Ethernet, the transmission requests are assumed to be Poisson in distribution, and the data is transmitted in packets at a ten megabit per second rate on the bus. The system is designed to be completely asynchronous utilizing a Manchester coding format. A preamble is used at the beginning of each packet for synchronization.

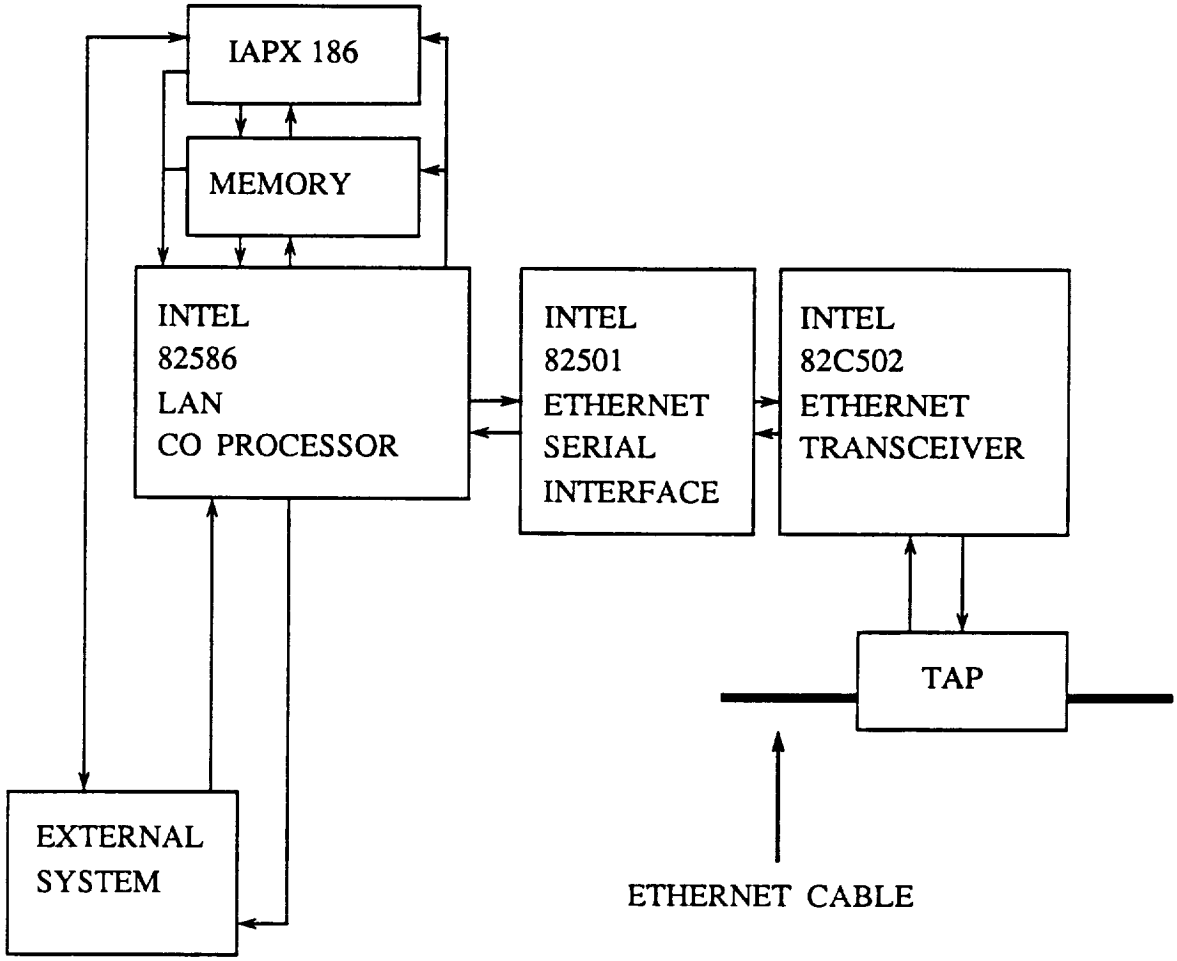


FIGURE 3.1 Ethernet Interconnection

3.1.1 ETHERNET PROTOCOL

The Ethernet original baseband version was designed, developed, and patented by Xerox and was publicly announced in 1979[DEC80]. Since then, a cooperative effort by Digital Equipment Corporation, Intel, and Xerox has produced an updated Ethernet which is considered the de facto standard for cable-based Local Area Networks because it is very close to the IEEE 802.3 CSMA/CD standard[IEEE82]. The Carrier Sense Multiple Access with Collision Detection (CSMA/CD) control technique is the more publicized method for bus/tree topologies and is compatible with the IEEE 802 standard[IEEE82].

Ethernet is a multi-access, packet-switched communications channel which is managed by the control technique CSMA/CD. It is used for carrying digital data among locally distributed computing systems. A primary goal of the Ethernet specification is compatibility. Ethernet was, in fact, the first to accomplish this by allowing devices from different manufacturers to communicate directly with one another.

Using the CSMA/CD control technique, each station attached to the bus must contend with other stations to access the bus. There is no central controller which allocates access to the channel. Each station must listen (i.e., use carrier sense) to determine whether the bus is free. A station must wait or defer its transmission until the bus is quiet if another station is transmitting. After gaining access to the bus, the transmitting station continues to monitor the medium to detect colliding transmissions on the bus. This is also called " listen while talk. " This leaves open the possibility that two stations will see the channel free at the same time and both will transmit into the supposedly open channel causing a collision. The resulting collision is then detected by both stations and both cease transmission. Each station then chooses a random backoff time as defined by the Exponential Backoff algorithm which

can be expressed by:

$$BP = R*T_S, \quad 0 < R \leq 2^k - 1$$

where:

BP = backoff period

R = a random number

k = the smaller of 8, or the number of collisions experienced so far during the current transmission

T_S = slot time, a time slightly greater than the round-trip propagation delay of the cable.

After each collision following the first, an Ethernet terminal will double the backoff range (the range of possible values of R) for eight contentions until it is two hundred and fifty-six times the slot time, and then leaves the backoff range at this value for the next eight contentions. If a terminal experiences sixteen contentions in trying to transmit a packet, it abandons the attempt and the packet is lost. Most Ethernet terminals will inform the host of this condition and allow the host's transport layer software to reschedule the packet or take other appropriate actions.

The station, after waiting the required backoff time, then waits until the bus is idle for 9.6 microseconds and then attempts to retransmit the packet. The Ethernet contention scheme works very well for systems that have Poisson distributed communications with an offered load of less than 3.33 megabits/second and continues to perform adequately for offered loads of up to 8.0 megabits/second. The major problem for this protocol is that no guarantee of data latency may be made. This is because a sta-

tions packet can continue to collide with other stations until the retry limit is reached. This lack of a guaranteed access to the bus is detrimental to the application of Ethernet to a real time system or to a system that has heavy bursts of time critical periodic traffic as may be seen in a data collection system.

3.1.2 DATA FORMAT AND STRUCTURE

Each station on the common coaxial cable must be able to transmit and receive packets with the packet format and spacing as shown in Figure 3.2. A packet is made up of various bytes with the last bit of each byte transmitted first, and the preamble beginning a transmission. A packet may not exceed 1526 bytes or be less than 72 bytes. Included in each of these numbers is: 8 bytes for the preamble, 14 bytes for the header, the data bytes, and 4 bytes for the CRC. The following defines each field of the frame:

1) Preamble: 64 bits alternating 1's and 0's, and ending with two consecutive 1's. The preamble is used by the receiver to establish bit synchronization and then to locate the first bit of the frame.

2) Destination Address: 48 bits specifying the station or stations which are to receive the packet. The packet may go to one station, a group of stations, or broadcast to all. This is determined by the first bit: a 0 indicates one destination, and a 1 indicates multiple stations. If all 48 bits are set to 1, then the packet is broadcast to all stations.

3) Source Address: 48 bits specifying the station which is transmitting the packet.

4) Type Field: 16 bits identifying the type of higher level protocol associated with the packet. This is used to interpret the following data field.

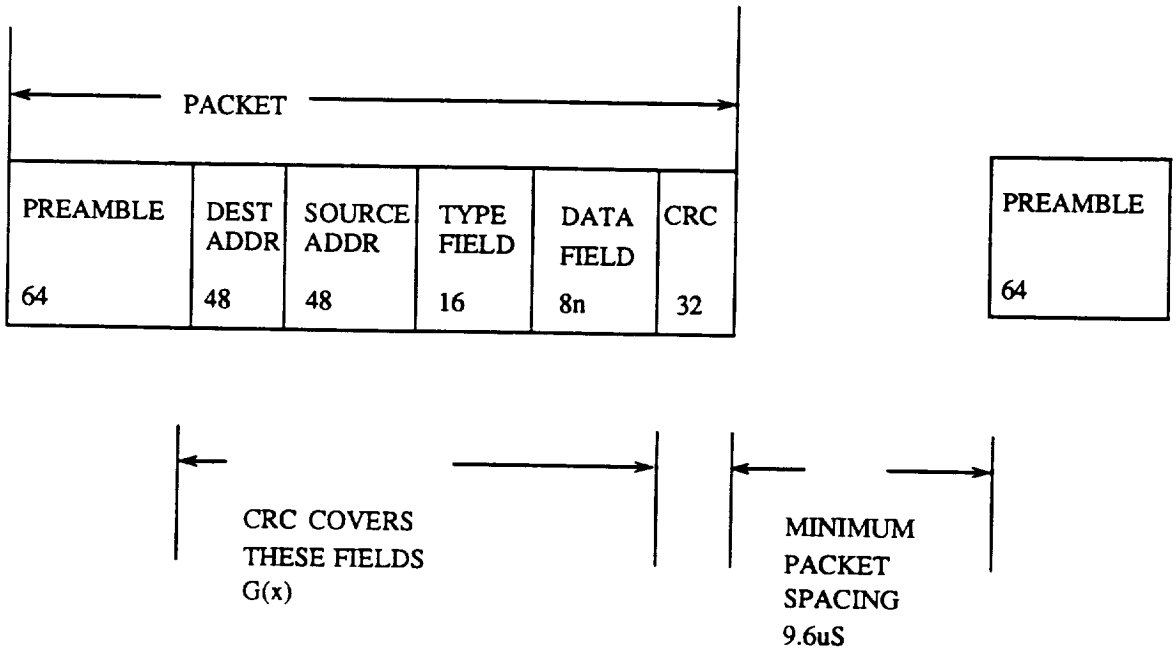


FIGURE 3.2 Ethernet Packet Structure

5) Data Field: 46 to 1500 bytes of data or pad characters. A minimum combination of 46 bytes to ensure that the frame will be distinguishable from a collision fragment.

6) CRC - Packet Check Sequence: 32 bits containing a redundancy check. The check is defined by the generating polynomial:

$$G(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1.$$

The CRC covers the address (destination/source), type, and data fields. The first transmitted bit of the destination field is the high-order term of the message polynomial to be divided by G(x) producing remainder R(x). The high-order term of R(x) is the first transmitted bit of the Packet Check Sequence field. The algorithm uses a linear feedback register which is initially preset to all 1's. After the last data

bit is transmitted, the contents of this register (the remainder) are inverted and transmitted as the CRC field. After receiving a good packet, the receiver's shift register contains:

$1100011100000100110111010111 1011 (x^{31}, ..., x^{0})$.

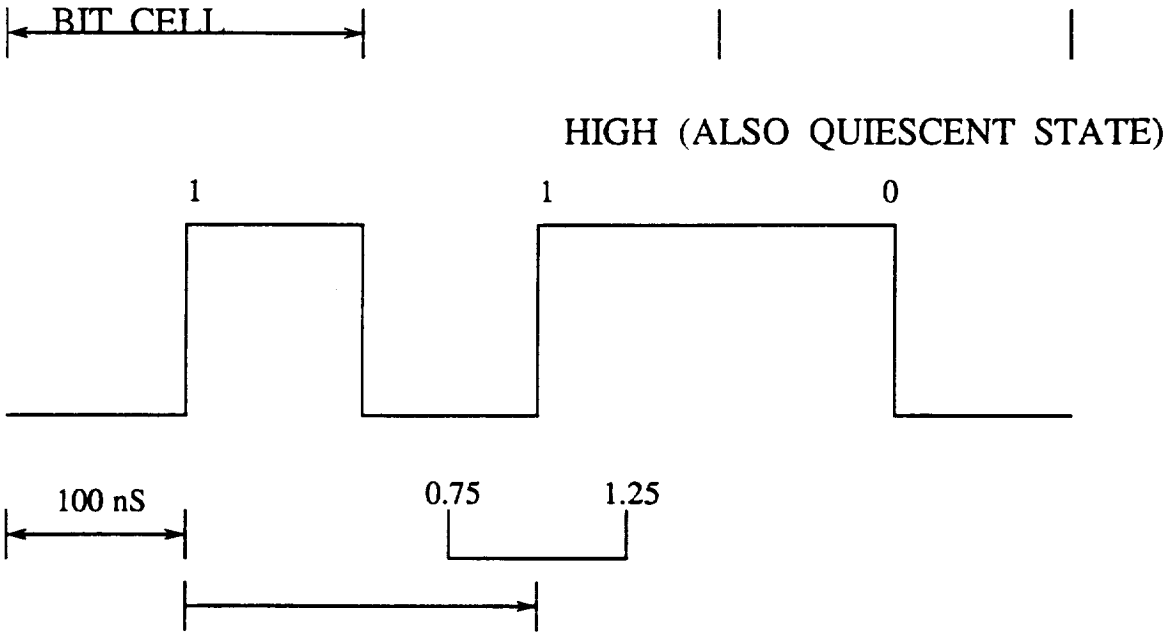
The Ethernet has an enforced waiting time on the bus of 9.6 microseconds, which is the minimum amount of time which must elapse after one transmission before another may begin. For one bit to travel from one end of the longest bus length allowed to the other (the round-trip propagation delay time) requires 51.2 microseconds. If any station receives a packet or bit sequence shorter than 72 bytes, the information is discarded and considered a collision fragment.

3.1.3 HARDWARE CHARACTERISTICS

The following section contains a brief overview of the hardware aspects of the Ethernet network system. Discussed here are: channel encoding, carrier detection, and the transceivers.

Manchester encoding is used on the coaxial cable. It has a 50% duty cycle and insures a transition in the middle of every bit cell (data transition). The first half of the bit contains the complement of the bit value while the second half contains the true value of the bit.

The presence of data transitions indicates that the carrier is present. If a transition is not seen between 0.75 and 1.25 bit times past the center of the last bit cell, then the carrier has been lost, indicating the end of a packet. For purposes of deferring transmission, the term carrier means any activity on the cable. Specifically, it is any activity on either receive or collision detect signals in the last 160 nanoseconds (see Figure 3.3).



LOGIC HIGH: 1= 0 mA = 0 VOLTS

LOGIC LOW: 0 = -82 mA = -2.05 VOLTS

CABLE HAS 0 VOLTS IN QUIESCENT STATE

FIGURE 3.3 Determination of Carrier at Receiver

At each station using the network, there are cables with taps which connect to a transceiver. The transceiver receives all signals on the cable, but only those addressed to it are accepted for action. The transceiver also is the device which transmits signals sufficiently strong to propagate the information from one end of the cable to the other. That is, every transmission on the cable will reach each transceiver.

The transceiver is designed so that if it fails, the faulty device does not jam or pollute the Ethernet cable. The devices are also built simply and are relatively inexpensive so that replacement of failed parts may be accomplished quickly. If a transceiver fails, it disconnects itself from the cable. The transceiver also contains a watchdog timer circuit which detects incorrect behavior and shuts down the transmitter in this event. The maximum number of stations which may be attached to the cable is 1000 stations spaced at least 2.5 meters apart to reduce the chance that objectionable standing waves will result.

Each Ethernet interface will require software and hardware. General interface software is usually available from the manufacturer but additional code is required to tailor the system to the specific application. The hardware required to implement an Ethernet terminal is a controller chip, a serial interface chip, a transceiver chip, transceiver cable, coaxial cable and a special cable TAP as illustrated in Figure 3.1. The estimated cost of components for a standard terminal is (10 MHz plastic chips):

PC BOARD and MISC	$100.00
MICROPROCESSOR	442.00
MEMORY 16K RAM	200.00
4 27C64 ROM	80.00
INTEL 82586	88.00
INTEL 82501	50.00
INTEL 82C502	43.00
TAP	290.00
TRANSCEIVER CABLE	52.00
COAXIAL CABLE	25.00
Terminal Cost:	$1370.00

3.2 MIL-STD-1553B (COMMAND/RESPONSE BUS)

MIL-STD-1553B establishes requirements for digital, command/response, time division multiplexing (Data Bus) techniques. It encompasses the data bus line and its interface electronics as illustrated in Figure 3.4. It also defines the concept of operation and information flow on the multiplexed data bus as well as the electrical and functional formats to be employed.

This section provides the reader who is not familiar with the MIL-STD-1553B protocol sufficient background to be able to follow the content of the remaining portions in this section. The Multiplex Application Handbook is used as reference for this section.[MILS87]

Because a Command/Response protocol is highly structured, unlike Ethernet, it is necessary to fully describe the protocol and the types of terminals allowed by MIL-STD-1553. A " terminal " in MIL-STD-1553 parlance is " The electronic mod-

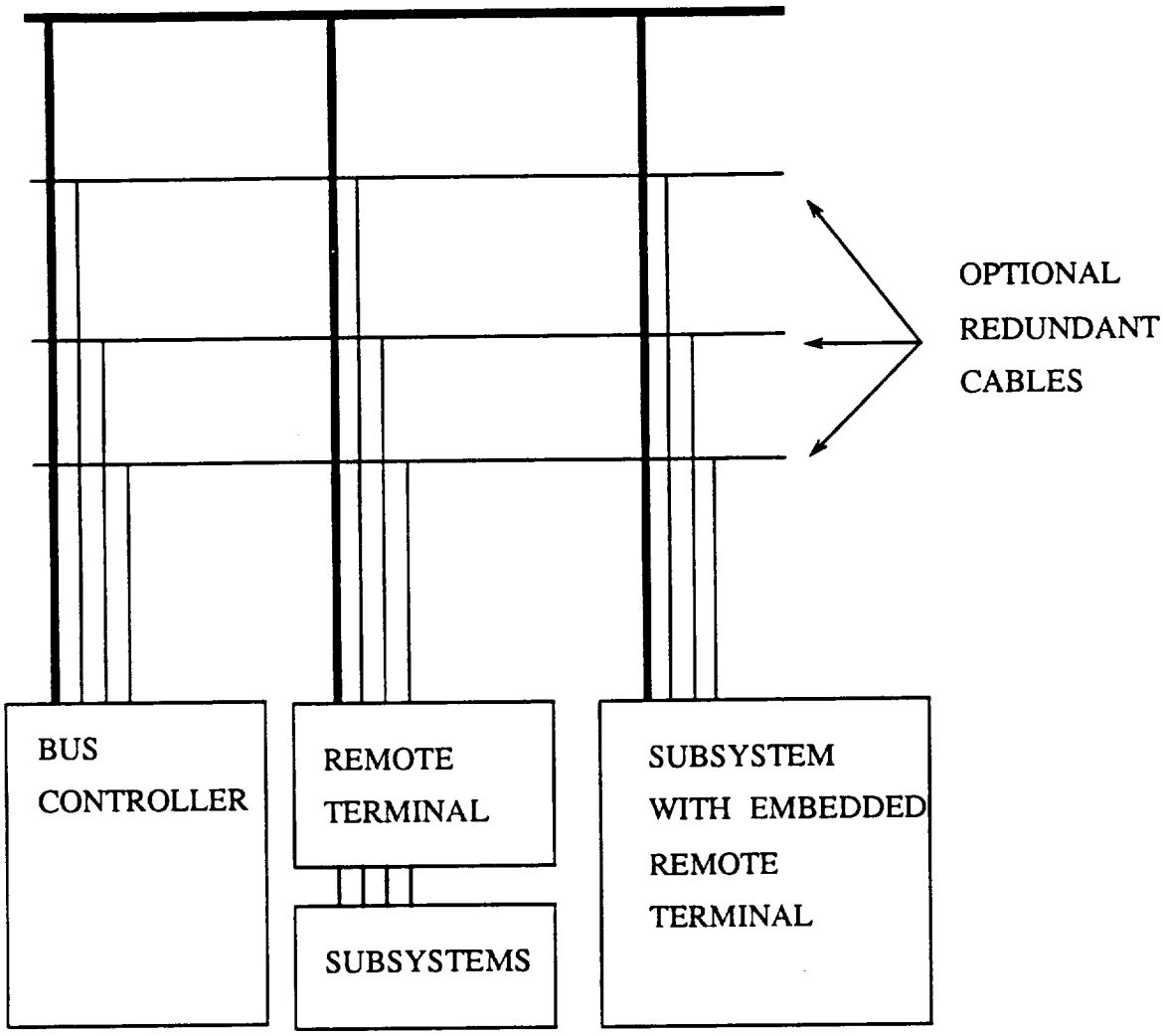


FIGURE 3.4 MIL-STD-1553B Basic Network Structure

ule necessary to interface the data bus with the subsystem and the subsystem with the data bus ... '' There are only three functional types of terminals: the bus controller, the bus monitor, and the remote terminal. The definition of the terminal as an electronic module should convey the notion of a unit that contains digital logic as a minimum and may frequently contain microprogrammed LSI or a microcomputer. A terminal serving as a bus monitor or bus controller must usually rely on a connection to and a dependence on a minicomputer for functional performance. Significant digi-

tal complexity is required because MIL-STD-1553B specifies response time and data storage requirements that require dedicated digital hardware. A possible hardware configuration that could be used to serve as either a bus controller, a bus monitor, or a remote terminal is presented in Figure 3.5.

3.2.1 THE BUS CONTROLLER

The bus controller is the terminal assigned the task of initiating information transfers on the data bus. Another requirement is that the bus controller have sole control of information transmission on the bus.

The MIL-STD-1553B defines a bus monitor as a terminal assigned the task of receiving bus traffic and extracting selected information to be used at a later time. Frequently, bus monitors are used for instrumentation. Any terminal that is neither a bus controller nor a bus monitor is a remote terminal.

MIL-STD-1553B is a serial digital bus. The data transfer on the bus is half duplex; that is, data may be transferred over the bus in either direction over a single line, but not in both directions on that line simultaneously. The transmission bit rate on the bus is 1.0 megabits per second. The data code is defined to be Manchester II bi-phase level. A logic one will be transmitted as a bipolar coded signal 1/0 (i.e., a positive pulse followed by a negative pulse). A logic zero will be a bipolar coded signal 0/1 (i.e., a negative pulse followed by a positive pulse). A transition through zero occurs at the midpoint of each bit time as shown in Figure 3.6. Independent terminals connected to the bus operate asynchronously because of the use of an independent clock source in each terminal for message transmission. Decoding is achieved in the receiving terminals by use of clock information derived from the message.

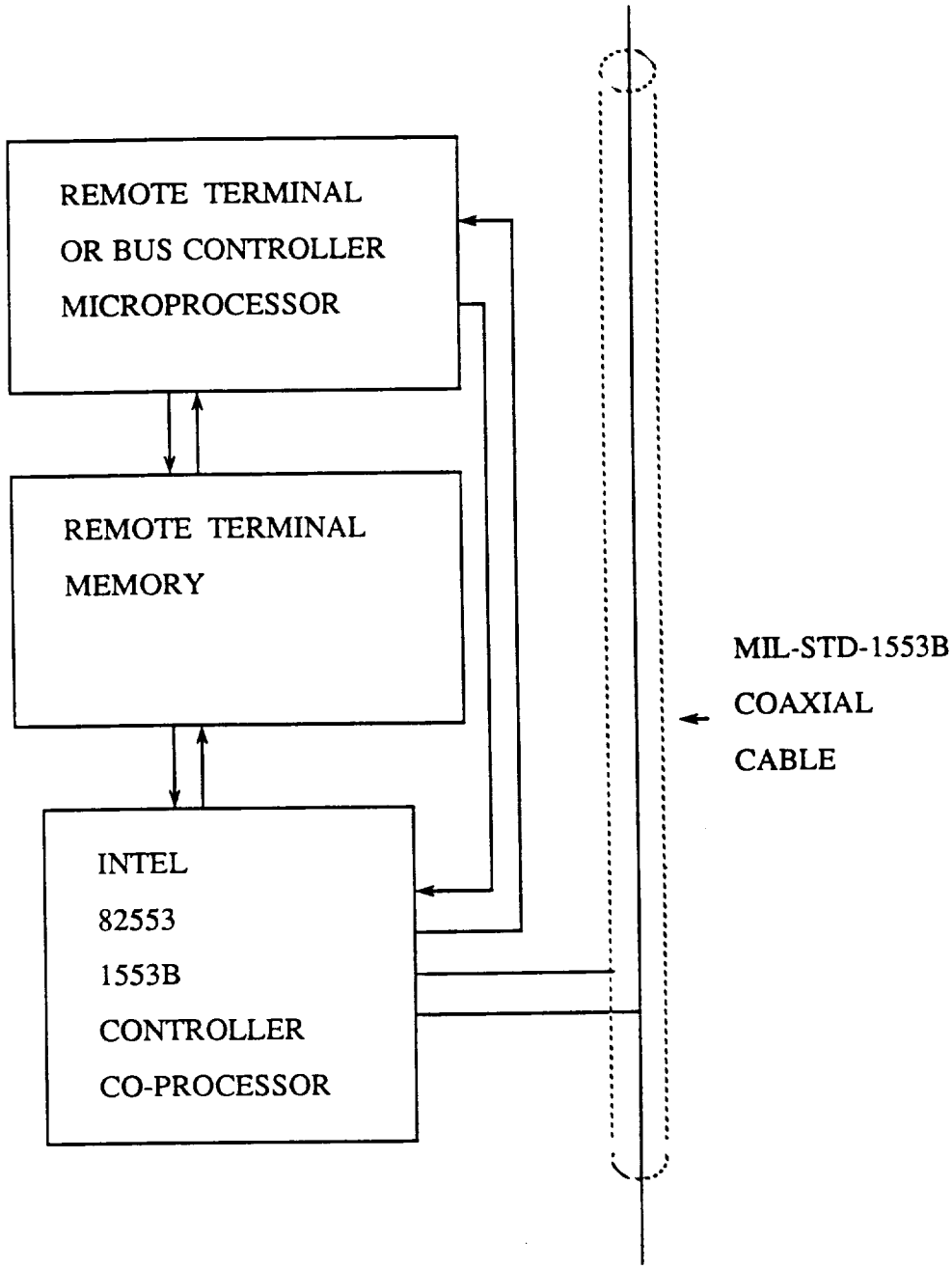


FIGURE 3.5 MIL-STD-1553B Remote Terminal/Bus Controller

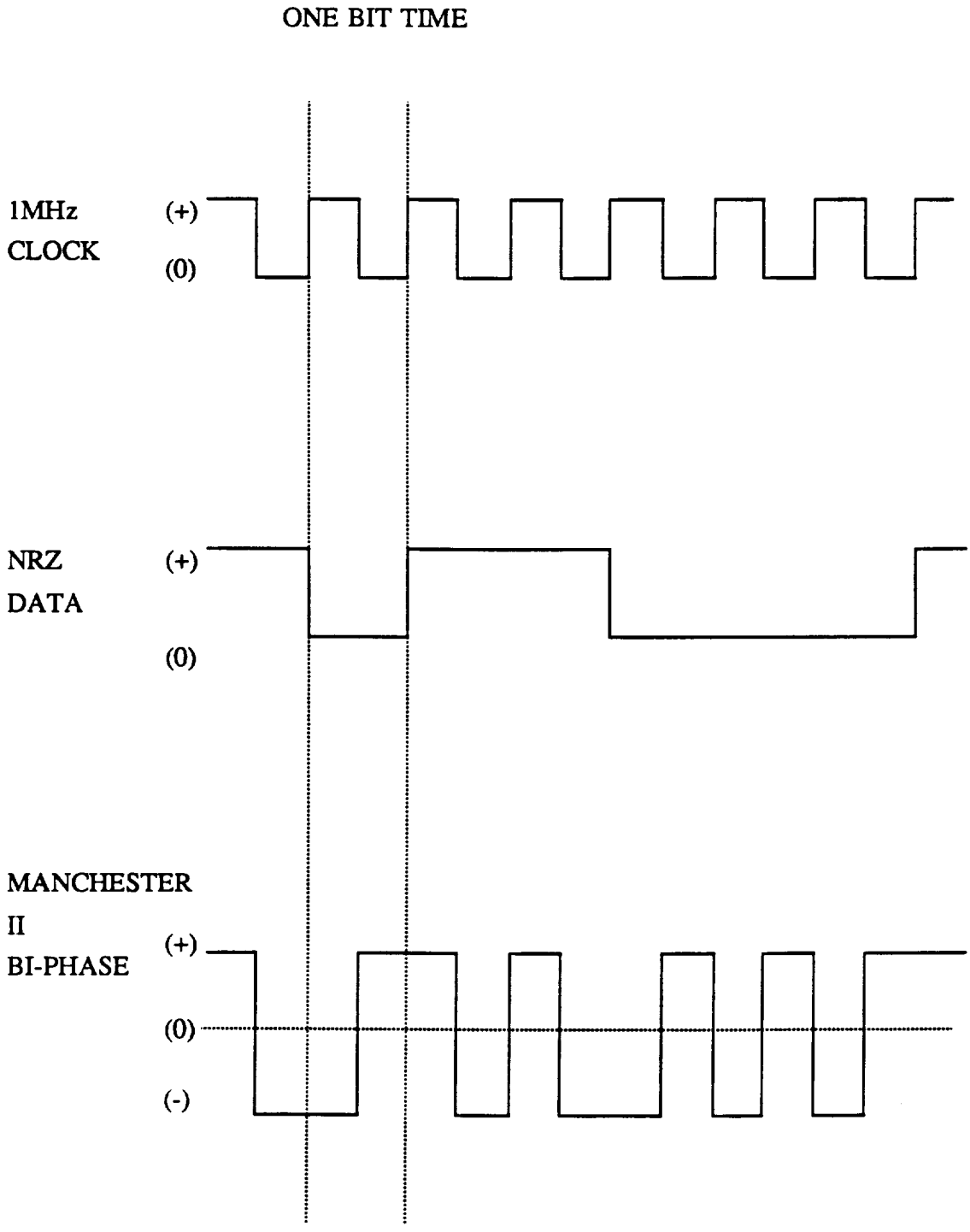


FIGURE 3.6 MIL-STD-1553B Manchester II Data Encoding

Data transfers over the bus are accomplished as a sequence of words. Each word is a sequence of 16 bits plus 3 bit times of sync and a parity bit for a total of 20 bit times. There are three types of words: command, status, and data as shown in Figure 3.7.

3.2.2 WORD TYPES

a. Command Word

A command word is comprised of a sync waveform, remote terminal address field, transmit/receive (T/R) bit, subaddress/modefield, word count/mode code field, and a parity (P) bit as shown in Figure 3.7. The command sync waveform is an invalid Manchester waveform as shown in Figure 3.7. The width is three bit times, with the sync waveform being positive for the first one and one-half bit times and then negative for the following one and one-half bit times.

The next five bits following the sync is the RT address. Each RT must be assigned a unique address. Decimal address 31 (11111) will not be assigned as a unique address. In addition to its unique address, an RT will be assigned the address 31 (11111) as a common address if the broadcast option is used. Each remote terminal is responsible for responding when its unique address is transmitted as part of a command word on the data bus by the active bus controller.

The next bit following the remote terminal address is the T/R bit, which indicates the action required of the RT. A logic zero indicates the RT is to receive while a logic one indicates the RT is to transmit.

BIT TIMES

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

COMMAND WORD

	REMOTE TERMINAL ADDRESS 5	T/R 1	SUBADDRESS/MODE 5	DATA WORD COUNT MODE CODE 5	P 1

T/R: TRANSMIT/RECEIVE
P1: PARITY

DATA WORD

	DATA 16	P 1

STATUS WORD

	REMOTE TERMINAL ADDRESS 5	ME 1	I 1	SR 1	RESERVED 3	BC 1	BU 1	SF 1	DB 1	TF 1	P 1

ME: MESSAGE ERROR
BC: BROADCAST COMMAND RECEIVED
I: INSTRUMENTATION
BU: BUSY
SR: SERVICE REQUESTS
F: SUBSYSTEM FLAG
DB: DYNAMIC BUS CONTROL ACCEPTANCE
TF: TERMINAL FLAG

FIGURE 3.7 MIL-STD-1553B Word Format

The next five bits following the T/R bit are utilized to indicate an RT subaddress or use of optional mode control. The subaddress/mode values of 00000 and 11111 are reserved for optional mode control. The mode code control is used only to communicate with the multiplex bus related hardware and to assist in the management of information flow. It will not be used to extract data from or feed data to a functional subsystem. Optional subaddress/mode code of 00000 or 11111 will imply that the contents of the word count field are to be decoded as a five bit mode command.

The next five bits following the subaddress/mode control indicate the number of data words to be either sent out or received by the RT or the optional mode code as specified in the previous paragraph. A maximum of 32 data words may be transmitted or received in any one message block. All 1's will indicate a decimal count of 31 and all 0's will indicate a decimal count of 32. The last bit in the command word is used for parity over the preceding 16 bits; odd parity is utilized.

b. Status Word

A status word is comprised of a sync waveform, RT address, message error bit, instrumentation bit, service request bit, three reserved bits, broadcast command received bit, busy bit, subsystem flag bit, dynamic bus control bit, terminal flag bit, and a parity bit as shown in Figure 3.7.

The status word sync waveform is the same as the command word sync. The next five bits following the sync contain the address of the RT which is transmitting the status word as indicated in Figure 3.7.

The status word bit at bit time nine (Figure 3.7) is utilized to indicate that one or more of the data words associated with the preceding receive command word from

the bus controller has failed to pass the RT's validity tests. The validity tests are explained later in this section. A logic one indicates the presence of a message error while a logic zero shows its absence. The status word bit at bit time 10 (Figure 3.7) is reserved for the instrumentation bit and will always be a logic 0. This bit is intended to be used in conjunction with a logic one in bit time 10 of the command word to distinguish between a command word and a status word. The status word bit at bit time 11 (Figure 3.7) is reserved for the service request bit. The use of this bit is optional. This bit, when used, indicates the need for the bus controller to take specific pre-defined actions relative to either the RT or associated subsystem. Multiple subsystems, interfaced to a single RT, which individually require a service request signal will logically OR their individual signals into the single status word bit. In the event this logical OR is performed, the designer must make provisions in a separate data word to identify the specific requesting subsystem. The service request bit is intended to be used only to trigger data transfer operations which take place as an exception rather than on a periodic basis. A logic one will indicate the presence of a service request and a logic zero its absence. If this function is not implemented, the bit should be set to zero.

The status word bits at bit times 12 through 14 are reserved for future use and should not be used. These bits should be set to a logic zero.

The status word bit at bit time 15 is set to a logic one to indicate that the preceding valid command word was a broadcast command while a logic zero indicates it was not a broadcast command. If the broadcast command option is not used, this bit should be set to a logic zero.

The status word bit at bit time 16 (Figure 3.7) is reserved for the busy bit.

The use of this bit is optional. This bit, when used, indicates that the RT or subsystem is unable to move data to or from the subsystem in compliance with the bus controller's command. A logic one will indicate the presence of a busy condition and a logic zero the absence of a busy condition. In the event the busy bit is set in response to a transmit command, then the RT will transmit its status word only. If this function is not implemented, the bit should be set to logic zero.

The status word bit at bit time 17 (Figure 3.7) is reserved for the subsystem flag bit. The use of this bit is optional. This bit, when used, will flag a subsystem fault condition and alert the bus controller to potentially invalid data. Multiple subsystems, interfaced to a single RT, which individually require a subsystem flag bit signal will logically OR their individual signals into the single status word bit. In the event this logical OR is performed, then the designer must make provisions in a separate data word to identify the specific reporting subsystem. A logic one indicates the presence of the flag, while a logic zero indicates its absence. If not used, this bit should be set to logic zero.

The status word bit at bit time 18 (Figure 3.7) is reserved for the acceptance of dynamic bus control. This bit is used if the RT implements the optional dynamic bus control function. A logic one will indicate acceptance of control and a logic zero will indicate rejection of control. If this function is not used, this bit should be set to logic zero.

The status word bit at bit time 19 (Figure 3.7) is reserved for the terminal flag function. The use of this bit is optional. This bit, when used, will flag an RT fault condition. A logic one indicates the presence of the flag, and a logic zero, its absence. If not used, this bit should be set to logic zero.

The last bit in the status word is used for parity over the preceding 16 bits. Odd parity will be utilized.

c. Data Word

A data word is comprised of a sync waveform, data bits, and a parity bit as shown in Figure 3.7. The data sync waveform is an invalid Manchester waveform as shown in Figure 3.7. The width is three bit times, with the waveform being negative for the first one and one-half bit times. Note that if the bits preceding and following the sync are logic ones, then the apparent width of the sync waveform is increased to four bit times.

The 16 bits following the sync are utilized for data transmission. The last bit in the data word (bit time 20, Figure 3.7) is used for parity over the preceding 16 bits. Odd parity is utilized. Data words are used to transmit information, control, and state data. Data words are distinguished from command and status words by the inverted three-bit sync pattern.

3.2.3 DATA TRANSFERS

a. Data Bus Operation

The multiplex data bus system will function asynchronously in a command/response mode, and transmission occurs in a half-duplex manner. Sole control of information transmission on the bus resides with the bus controller which will initiate all transmissions. The information flow on the data bus is comprised of messages which are, in turn, formed by three types of words (command, data, and status).

A single message is the transmission of a command word, status word, and data words if they are specified. For the case of a remote terminal to remote terminal (RT to RT) transmission, the message will include two command words, two status

words, and associated data words.

The messages transmitted on the data bus will be in accordance with the formats on Figure 3.8 and Figure 3.9. The bus controller is responsible for providing a minimum Intermessage gap of 4.0 microseconds(μsec) as shown on Figures 3.8 and 3.9. This time period is measured from the mid-bit zero crossing of the last bit of the

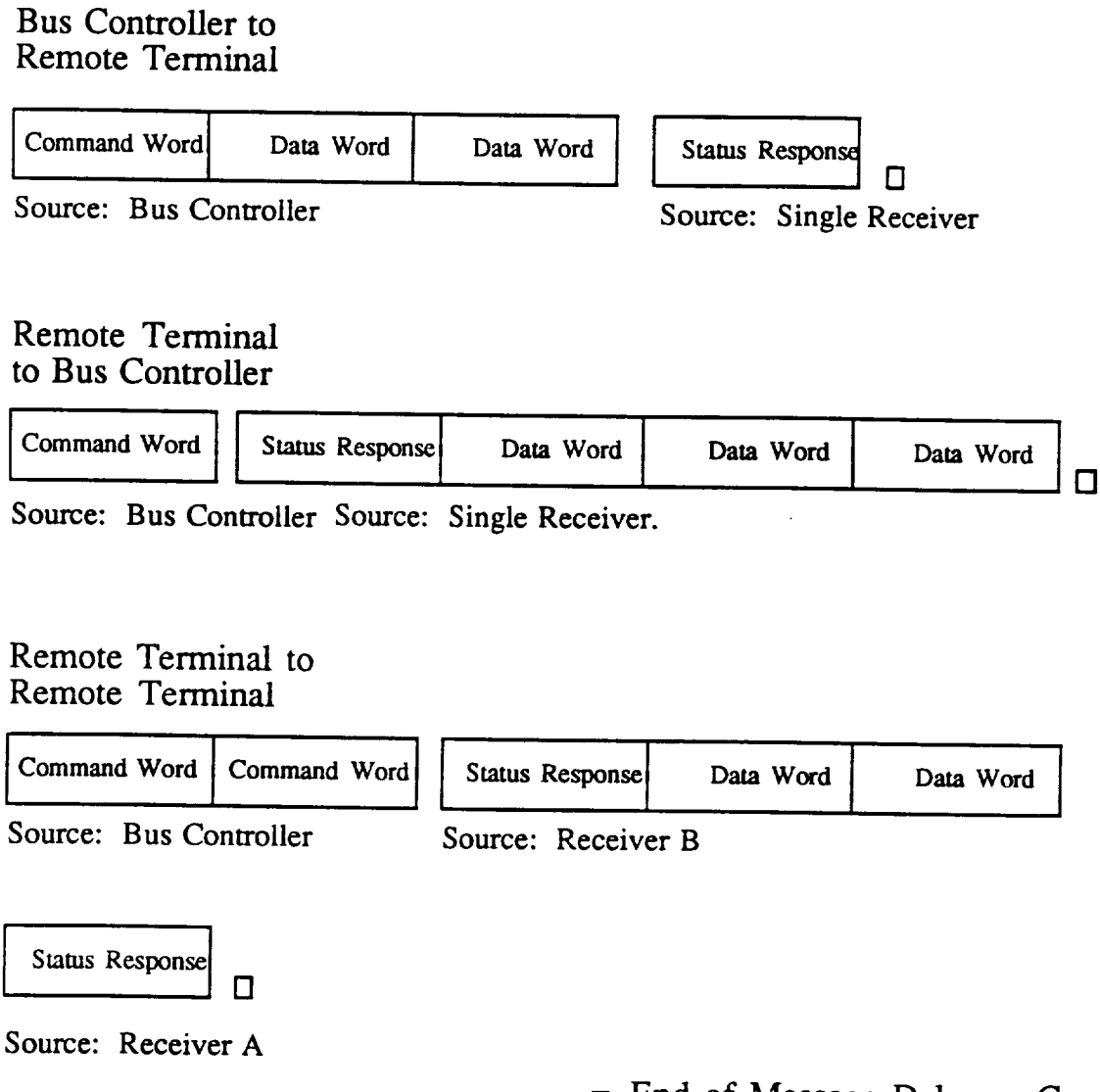
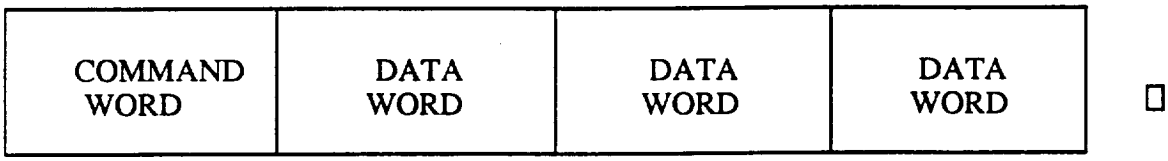


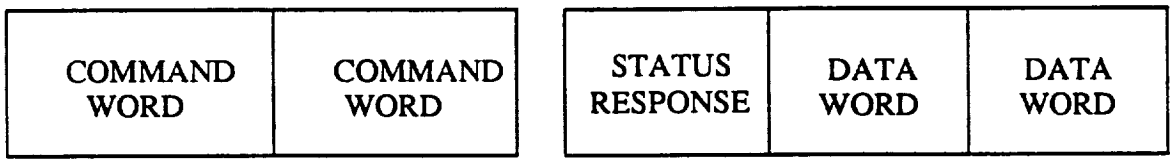
FIGURE 3.8 MIL-STD-1553B Receiver Data Message Format

BUS CONTROLLER TO REMOTE TERMINALS



SOURCE: BUS CONTROLLER
ADDRESS: 11111
SUBADRESS: UNIQUE = 11111 OR 00000
WORD COUNT: 1-32
T/R: RECEIVE

REMOTE TERMINAL TO REMOTE TERMINALS



SOURCE: BUS CONTROLLER
ADDRESS: 11111
SUBADRESS: UNIQUE = 11111 OR 00000
WORD COUNT: 1-32
T/R: RECEIVE

SOURCE: RECEIVER A

SOURCE: BUS CONTROLLER
ADDRESS: UNIQUE A
SUBADRESS: UNIQUE = 11111 OR 00000
WORD COUNT: 1-32
T/R: TRANSMIT

END OF MESSAGE DELAY OR GAP
RESPONSE TIME DELAY OR GAP

FIGURE 3.9 MIL-STD-1553B Multiple Receiver Data Message Format

preceding message to the mid-zero crossing of the next command word sync as shown in Figure 3.10. A remote terminal will respond to a valid command within the time period of 4.0 to 12.0 microseconds. This time period is shown as T on Figure 3.10. Different message formats transmitted on the bus are explained in the following paragraphs.

b. Bus Controller to Remote Terminal Transfers

The bus controller will issue a receive command followed by the specified number of data words. The RT will, after message validation, transmit a status word back

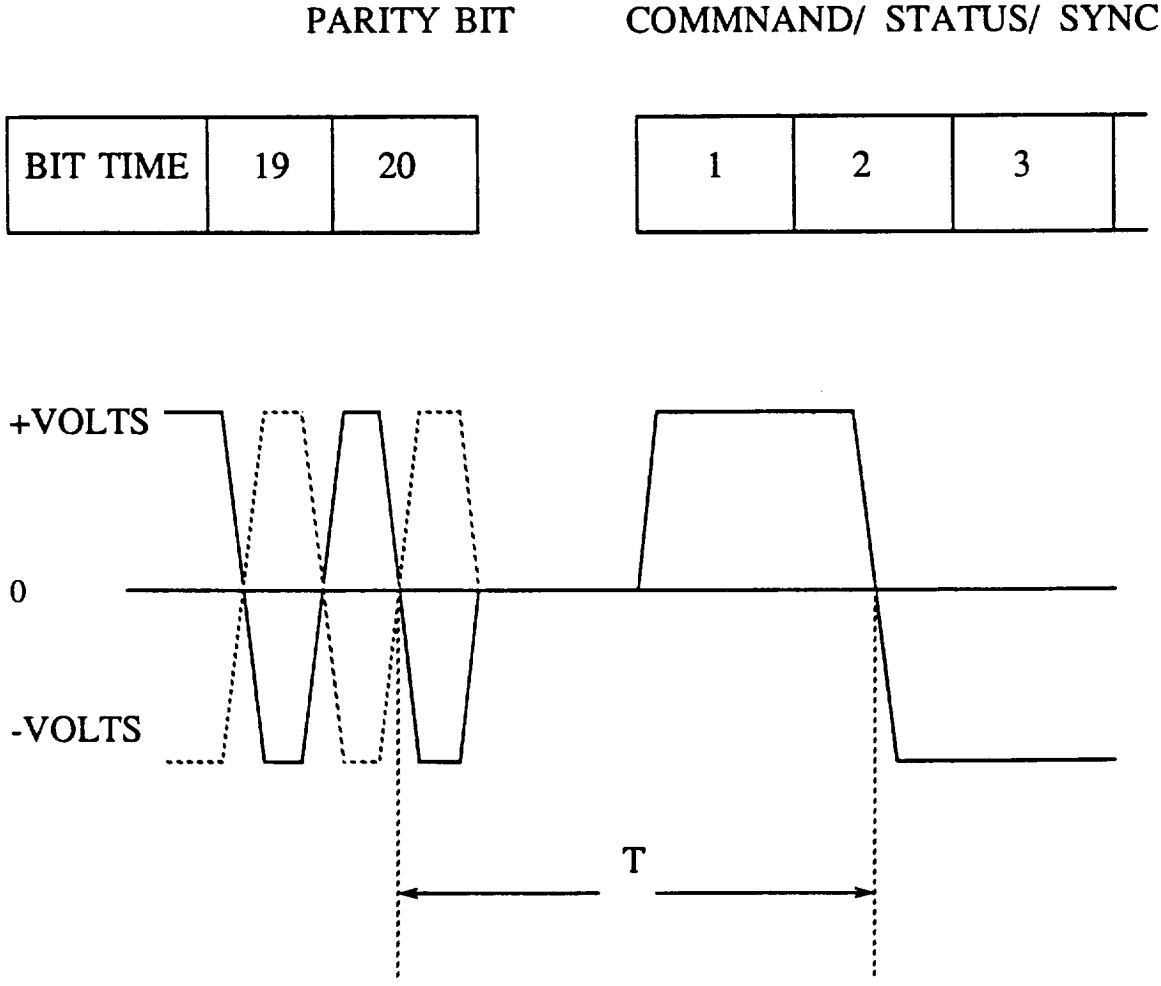


FIGURE 3.10 MIL-STD-1553B Intermessage Gap and Response Time

to the controller. The command and data words will be transmitted in a contiguous fashion with no interword gaps.

c. Remote Terminal to Bus Controller Transfers

The bus controller will issue a transmit command to the RT. The RT will, after command word validation, transmit a status word back to the bus controller, followed by the specified number of data words. The status and data words will be transmitted in a contiguous fashion with no interword gaps.

d. Remote Terminal to Remote Terminal Transfers

The bus controller will issue a receive command to RT A followed contiguously by a transmit command to RT B. RT B will, after command verification, transmit a status word followed by the specified number of data words. The status and data words will be transmitted in a contiguous fashion with no gap. At the conclusion of the data transmission by RT B, RT A will transmit a status word within the specified time period.

e. Bus Controller to Remote Terminal(s) Transfer (broadcast)

If the bus controller desires to transfer data to all remote terminals with the broadcast option, then the bus controller issues a receive command word with 11111 in the RT address field followed by the specified number of data words. The command word and data words are transmitted in a contiguous fashion with no gap. The RT(s) with the broadcast option will, after message validation, set the broadcast command received bit in the status word but will not transmit the status word.

f. Remote Terminal to Remote Terminal(s) Transfers (broadcast)

The bus controller may allow two remote terminals to communicate. To accomplish this the bus controller will issue a receive command word with 11111 in

the RT address field followed by a transmit command to RT A using the RT's address. RT A specifies the number of data words. The status and data words are transmitted in a contiguous fashion with no gap. The RT(s) with the broadcast option, excluding RT A and after message validation, will set the broadcast received bit in the status word but will not transmit the status word.

g. Error Rate

The terminal will exhibit a maximum word error rate of one part in 10^9 on all words received by the terminal, after validation of sync bits, bit count, Manchester II code and odd parity, when operating in the presence of additive white Gaussian noise distributed over a bandwidth of 1.0 kHz to 4.0 MHz at an RMS amplitude of 140 mV for transformer coupled buses. Any fault which causes the message error bit to be set in the terminal's status word, or one which causes a terminal to fail to respond to a valid command, is counted as a word error. It should be noted that this specification sets the upper bound on the word error rate. Typically, the error rates are an order or two of magnitude lower. A CRC check may be added in the form of additional data words at the end of a group of data words. However, the gain in performance would be minimal since parity checks are used on a word-by-word basis. An error correcting technique has been devised for MIL-STD-1553B.

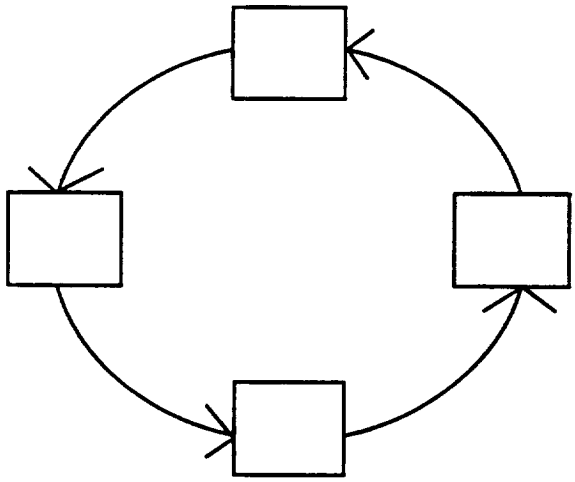
3.2.4 TERMINAL COST

Each terminal will require software and hardware. The cost of the hardware required to implement a typical Remote Terminal or Bus Controller is approximately:

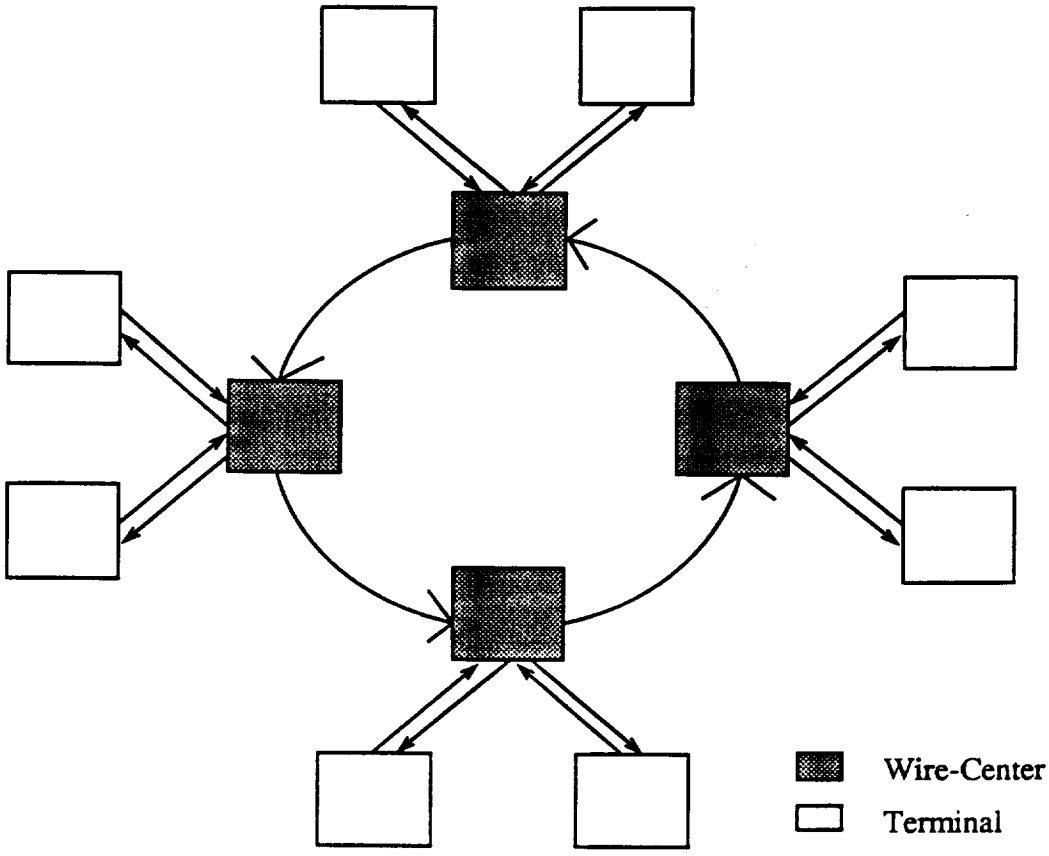
PC Board and Misc.	$100.00
1 Microprocessor Controller	442.00
1 Memory (16K)	280.00
1 1553 Transceiver Chip	650.00
1 Transformer Tap	35.00
Connection Cable	10.00
Terminal Cost	$1517.00

3.3 THE ProNET PROTOCOL

ProNet was developed by Proteon Inc. for use on the model p1200 Multibus LAN. The information in this section is based on information found in the "Operation and Maintenance Manual for the ProNet Model p1200 Multibus Local Network System" [PROT84]. ProNet operates on a classic ring or "star-shaped" ring, in which terminals are connected to the ring through a "wire center", which in the case of a terminal failure, can make the necessary connections to bypass that terminal, leaving the ring intact. These two configurations are shown in Figure 3.11. For additional reliablility, each ProNet ring is actually two counter-rotating rings, i.e., one ring in which data flows clockwise, and another in which data flows counterclockwise. If one ring should fail, communication can then take place on the other ring in a procedure known as "switch-back". If both rings should break at the same place (for example, a physical break or a terminal failure) then the terminals on either side of the break can go into "loop-back", that is, connect the counter-rotating rings into one large ring, bypassing the break. Counter-rotating rings, switch- and loop-back are illustrated in Figure 3.12. The basic terminal block diagram is shown in Figure 3.13.



"Classic" Ring



"Star-shaped" Ring

FIGURE 3.11 ProNET Ring Configurations

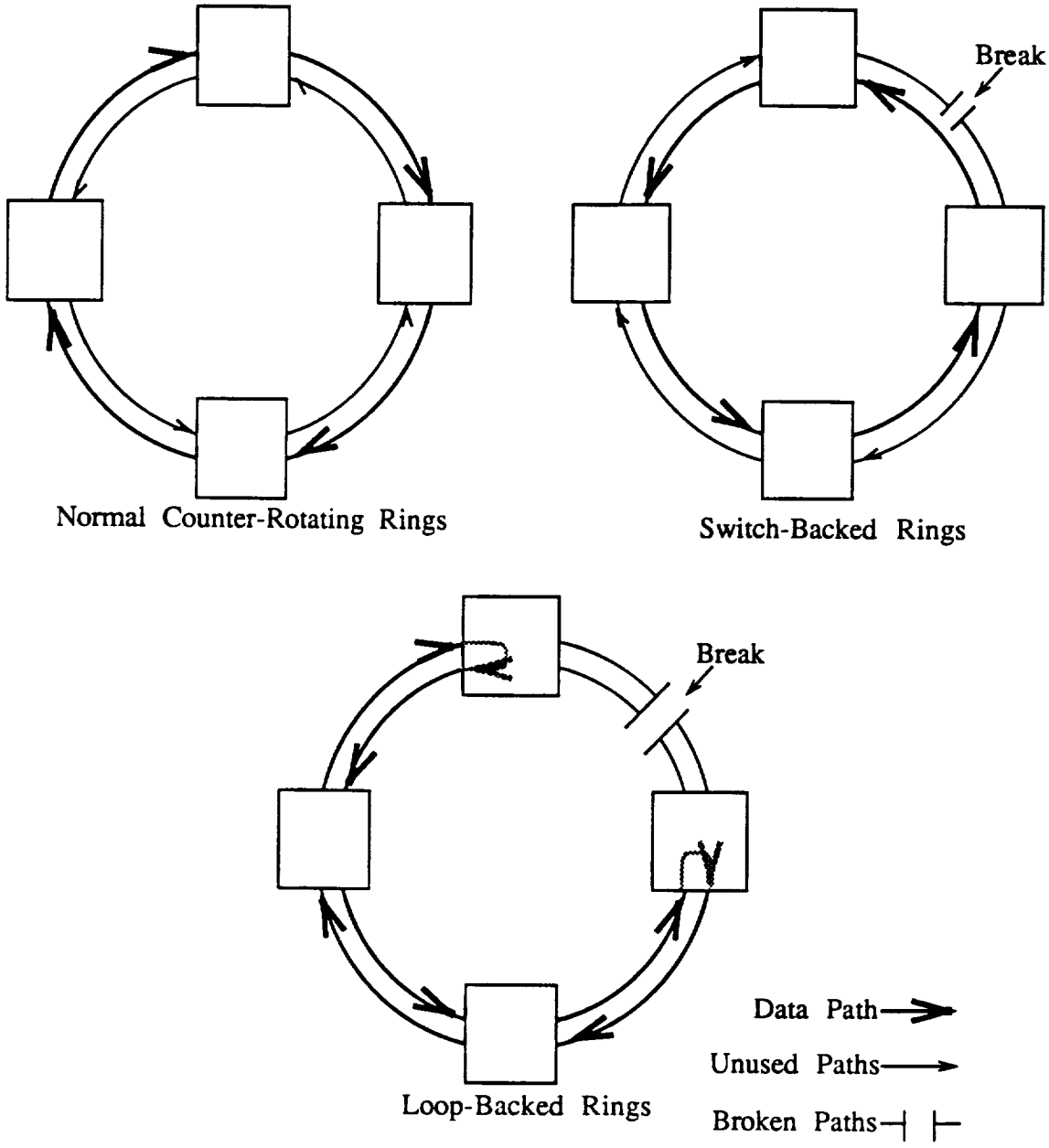


FIGURE 3.12 ProNET Ring Configurations

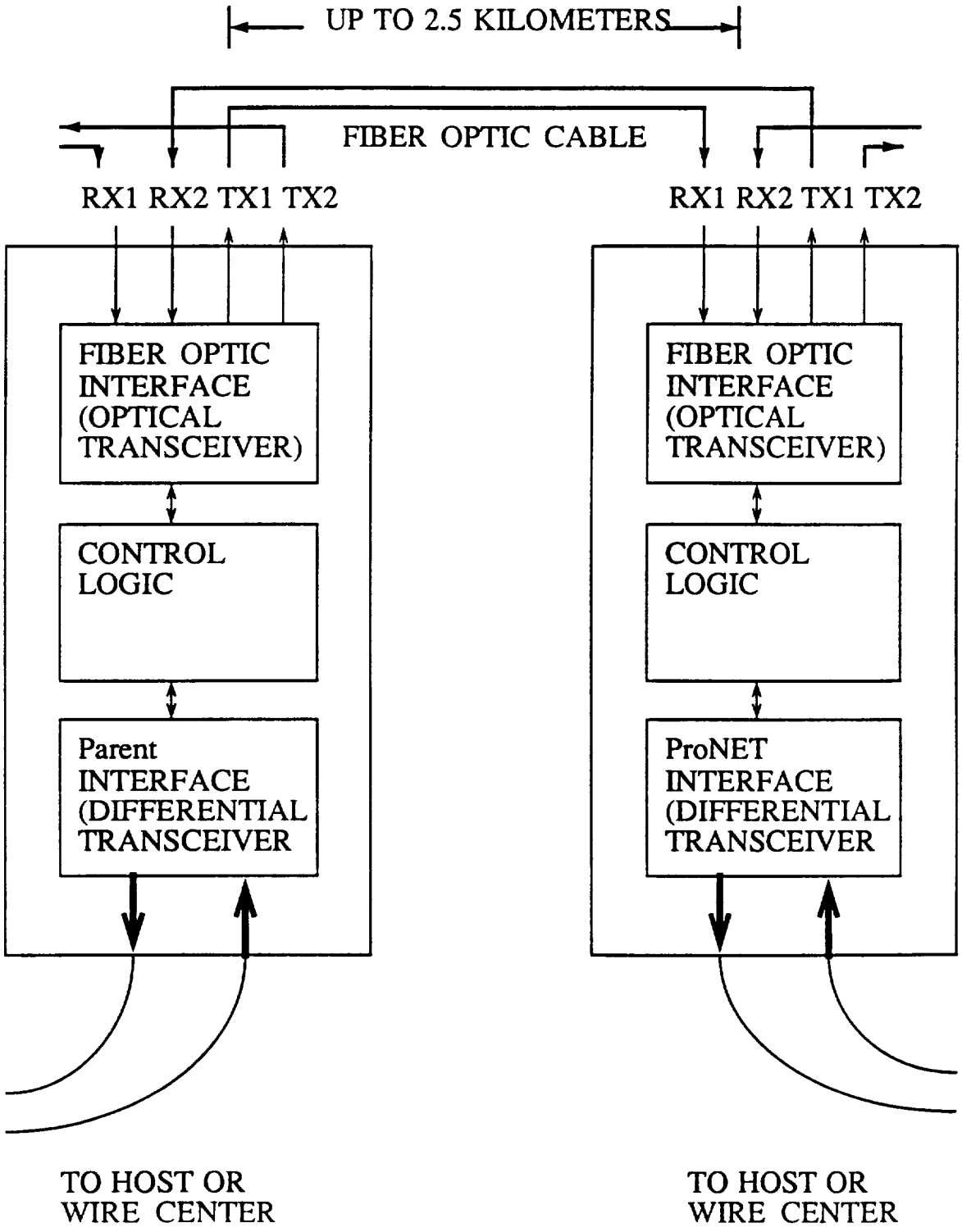


FIGURE 3.13 ProNET Terminal

3.3.1 ProNET CONTROL WORD AND MESSAGE FORMATS

As shown in Figure 3.14, the token consists of the "flag" (a zero followed by a string of seven ones), and two additional ones. When a terminal that has no data to send receives the token, it passes or "repeats" it to the next terminal in the ring.

If a terminal does have data to be transmitted, it changes the last bit of the token to 0, thus making it a Beginning of Message (BOM) character. It then adds eight bit source and destination address bytes, its data or "message" (up to 2044 eight-bit bytes), an End of Message (EOM) character consisting of the flag and an additional zero, a parity check bit, a message refused/accepted status bit, set initially to 1, and finally, a control character indicating the end of the packet, either a BOM of another packet, or the token. The packet format is shown as Figure 3.15.

In order to assure that no control character occurs in the message data, ProNet employs "bit stuffing." While creating a packet for transmission, if a terminal detects a stream of six consecutive ones, it "stuffs" a zero into the data behind them, insuring that the seven 1's of the flag will never occur in the data message. This stuffed zero will be removed by the addressed terminal as it "copies" the data from the ring into a buffer for its own use.

When a terminal recognizes its address as the destination address in a data packet, it copies the message part of the packet into a buffer for its own use, and then sends the packet back onto the ring, resetting the message refused/accepted status bit to zero. If a terminal is for some reason unable to copy data addressed to it, it repeats the packet along the ring, leaving the message refused/accepted bit set at 1. When a terminal detects a data packet which it had previously transmitted, it removes it from the ring, leaving only the token or the BOM of the next message.

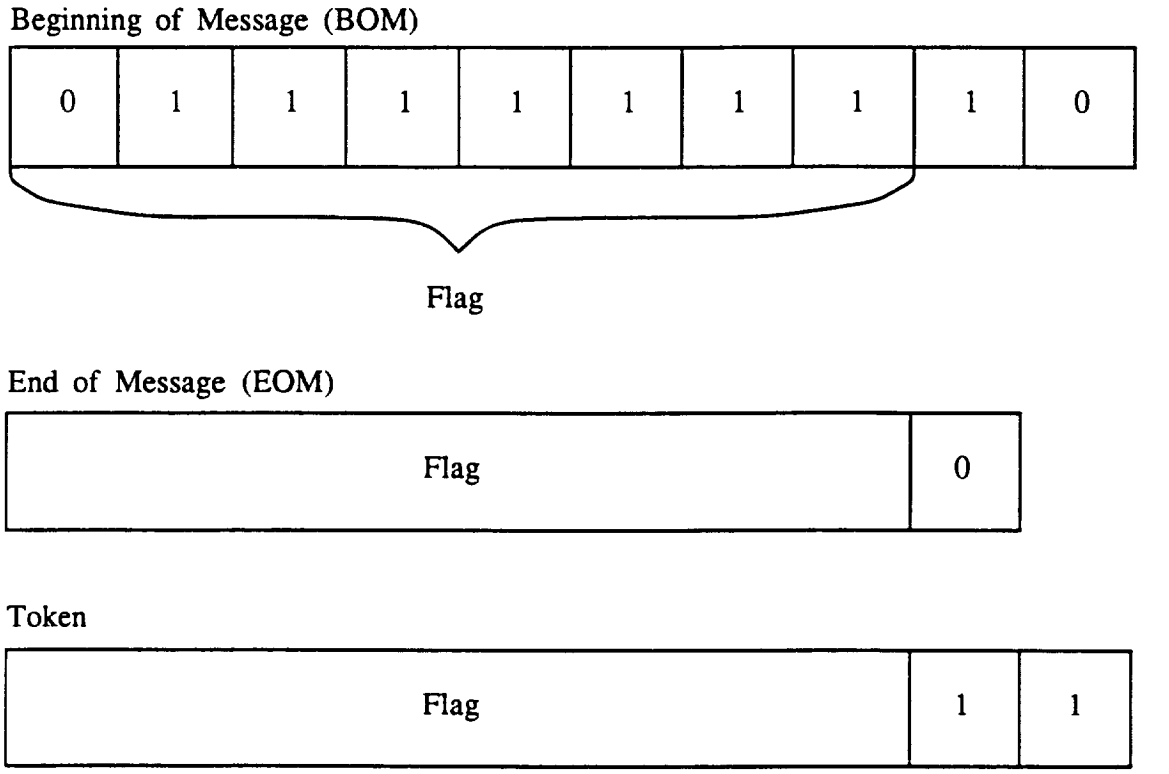


FIGURE 3.14 ProNET Control Character Format

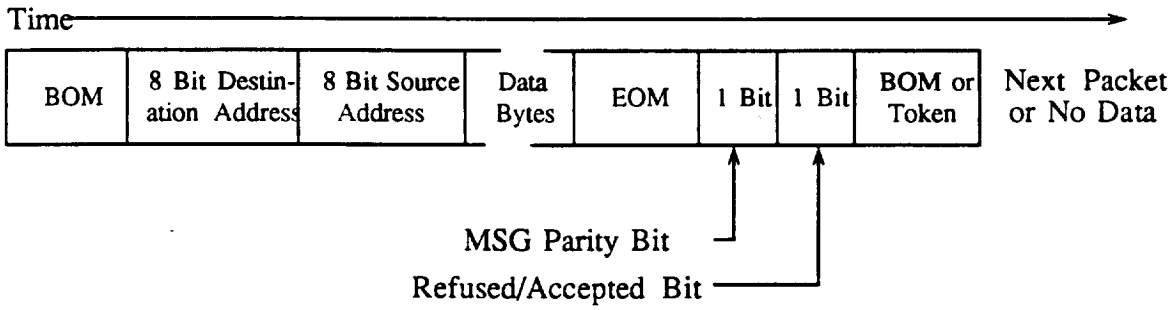


FIGURE 3.15 ProNET Packet Format

Depending on the application, the terminal may monitor the message refused/accepted bit and retransmit its data (if necessary) upon again receiving the token.

In order for the ring to recover from errors such as the loss of a token or packet, each terminal is equipped with three hardware timers: the token timer, the flag timer, and the message lost timer. The token and flag timers track the amount of time between the detection of a token or the flag respectively. If either of these counts exceed a set amount, the terminal will "re-initialize" the ring by generating and repeating a token. If two or more terminals try to re-initialize the ring within 500 μseconds of each other, a collision will occur and the ring will still be without a token. Upon the detection of a collision, the token and flag timers will be reset, leaving it for another terminal to re-initialize the ring. Proteon Inc. has determined that collisions during re-initialization are very unlikely.

The message lost timer serves a different function. After a terminal has transmitted a packet, it repeats no other data along the ring until it detects its own packet which indicates that the packet has made it successfully around the ring. This is to remove "noise" from the ring. The message lost timer begins counting when a packet is transmitted and resets when that same packet is again detected. If the message lost timer exceeds a set count (determined by the maximum amount of time required for a packet to entirely circle the ring), it is assumed that the packet is lost and the terminal resumes repeating all data that comes into it. Depending on the application, the terminal may or may not attempt to retransmit the lost packet upon receiving a token.

3.4 THE HYPERchannel PROTOCOL:

The HYPERchannel protocol is, like Ethernet, a CSMA protocol. This means that each terminal "listens" to the medium, in this case a bus, and transmits only

when the bus is free. HYPERchannel is unlike Ethernet, in that it does not employ collision detection. Instead, it uses message acknowledgments, several delay types, and prioritized terminals to ensure that only very short messages will experience a collision.

3.4.1 HYPERchannel DELAY SEQUENCE

In order to avoid collisions, after the bus becomes idle, HYPERchannel employs a sequence of delays during which only certain terminals may transmit. That sequence of delays is as follows:

a) Fixed Delay

During this time, the terminal which received the previous transmission sends a response frame (to be described later) to the terminal from which it received the transmission. All other terminals experience a delay, whose length is described by the equation:

Fixed Delay = 4 nanoseconds x (trunk length in feet) + 2.08 microseconds

which is slightly greater than twice the amount of time it takes for a signal to propagate the entire length of the bus and back.

b) N-Delay

In HYPERchannel, each terminal is assigned a unique priority used in determining the amount of time it must wait after the fixed delay before it may transmit, or its N-Delay. The N-Delay of each terminal may be expressed by the following equations:

$$\text{N-Delay}(K) = \text{N-Delay}(K-1) + 4 \text{ nanoseconds} \times d + 1.6 \text{ microseconds}, \quad K = 1,2,\ldots,L$$
$$\text{N-Delay}(1) = 4.8 \text{ microseconds}$$

where

K = priority index of the terminal
L = the number of terminal on the bus

d = the distance in feet from the terminal of priority K-1 to the terminal of priority K

Thus, each terminal is guaranteed a period of 1.6 μseconds in which it may initiate a transmission guaranteed to be without collision. It is important to note that terminals with low priority indexes are able to transmit more often than terminals with high priority indexes. The times in which terminals are guaranteed collision-free transmission are collectively referred to as the scheduling period.

c) End Delay

After the scheduling period, each terminal must wait an additional time period before it may begin transmission. During this time, the terminal listens to the bus medium to insure that the terminal with the lowest priority, that is, the terminal with priority index, L, has not begun a transmission. This listening period is referred to as the end delay, and its length for each terminal may be expressed by the equation:

End Delay(K) = N-Delay(L) + 4 nseconds x d′ + 1.6 μseconds, K=1,2,...,L-1

where L and K are defined the same as for N-Delay, but d′ is defined as the distance in feet from the terminal of priority index K to the terminal farthest from it.

After the end delay comes the contention period in which any terminal may transmit if it senses that the bus is idle. This is the only time when collisions can occur. HYPERchannel terminals do not detect collisions but instead rely on an acknowledgment from the terminal to which their transmission was addressed during the fixed delay period. If this acknowledgment does not arrive, they know their transmission was unsuccessful.

3.4.2 THE WAIT FLIP-FLOP

To prevent high-priority terminals from dominating the bus medium, each terminal is equipped with a wait flip-flop that is set when a terminal completes a trans-

mission and cleared at the beginning of the contention period. A terminal may not transmit when its wait flip-flop is set. The wait flip-flop of any individual terminal may be disabled if necessary, depending on its application.

3.4.3 HYPERchannel FRAMES AND SEQUENCES

The smallest unit of data transmitted by a HYPERchannel terminal is called the frame. There are three types of frames: transmission, response, and message-and-data frames. The function of each is described below:

a) Transmission Frames

Transmission frames are used for "handshaking", i.e., the exchange of control and status information between two terminals. The terminology used in this report often differs from that used by Network Systems Corporation, the manufacturers of HYPERchannel systems. When this is the case, the equivalent Network Systems Corporation term will be placed in parentheses after the given term. The transmission frames are listed and described below:

1) Request Status, RS (Copy Registers): The request status frame is used by a terminal to see if a terminal to which it wishes to transmit is capable of receiving the transmission. In frame sequences, the request status frame captures the bus for the transmission of subsequent frames.

2) End Message Proper, EMP (Clear Flag 8): This frame is sent by a transmitting terminal to indicate to the receiver that it has completed a message proper frame (to be described below). Flag 8 in the receiver is set when it is waiting for a message proper and cleared by the transmitting terminal when the complete message proper has been transmitted.

3) Prepare for Associated Data, PAD (Set Flag A): This frame alerts the receiving terminal that after the message proper, associated data frames (to be explained below) will follow.

4) Ready for Associated Data, RAD (Clear Flag 9): After a terminal receives a PAD frame, it must prepare buffer space in order to receive the associated data. When it has done this, it sends a RAD frame to the terminal which sent it the PAD. A terminal will not transmit associated data frames to their destination terminal until the destination terminal sends it a RAD.

5) End of Associated Data, EAD (Clear Flag A): This frame is transmitted by a terminal to indicate to the receiver that there will be no more associated data frames.

6) Request Virtual Circuit, RVC (Set Reserve Flag): This frame is sent by a terminal when it requires a virtual circuit connection (to be described later) with the receiving terminal.

b) Response Frames

Response frames are transmitted only during the fixed delay to acknowledge the reception of a frame. A response frame may contain status information if it is used to acknowledge a request status frame.

c) Message and Data Frames

There are two types of message and data frames: (1) message proper frames which contain up to 64 bytes of data, and (2) associated data frames which may contain up to 2 kilobytes of data.

The transmission of data from one terminal to another requires the exchange of several frames known as a frame sequence. Terminals do not control the bus for the entire duration of the frame sequence but rather relinquish control at various times as shown in Figures 3.16 and 3.17. There are two types of frame sequences: (1) message only sequences, in which only the data contained in a single message proper is exchanged, and (2) message with data sequences in which associated data frames follow a message proper.

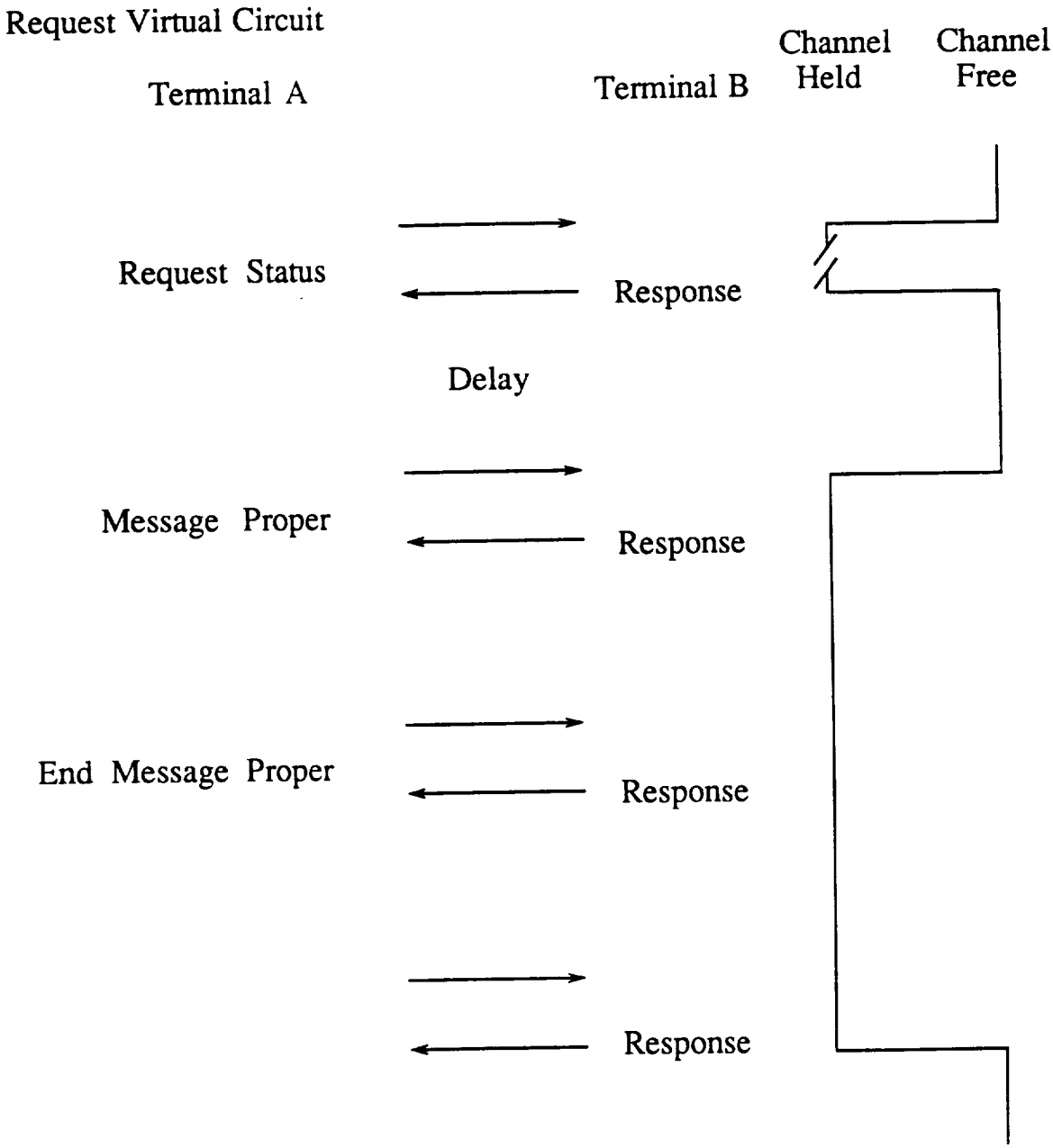


FIGURE 3.16 Timing for Transfer of Message-Only Sequence from Terminal A to Terminal B

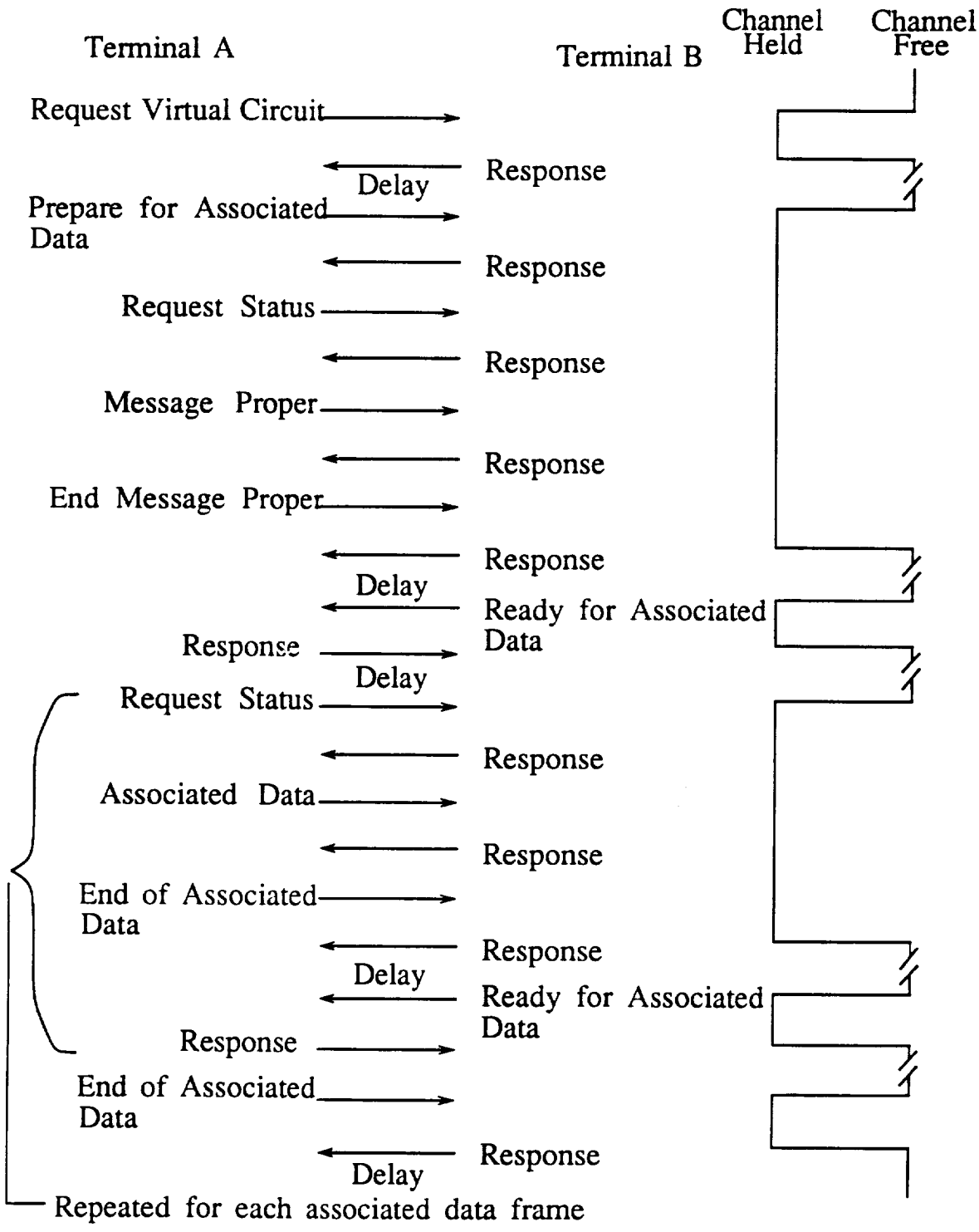


FIGURE 3.17 Timing for Transfer of Message-With-Data Sequence from Terminal A to Terminal B

In order for two terminals to exchange a message with data sequence, they must establish a virtual circuit. The sending terminal, from which the data originates, reserves itself to communicate only with the receiving terminal, to which the data is destined. It then sends a request virtual circuit frame to the receiving terminal. If that terminal is capable of receiving a data sequence, it will reserve itself to communicate only with the sending terminal. When the two terminals are reserved to communicate only with each other, they are in a virtual circuit connection.

If the receiving terminal is unable to make a virtual circuit connection, the sending terminal will wait for a delay period which can be expressed by the equation

$$\text{Retry Delay (k)} = 2^{(k-1) \text{ modulo } 7} \times 1 \text{ μsecond, } k=1,2,...,RC$$

where k is the number of the attempt and RC is a terminal parameter known as the Retry Count. Thus, after failing to establish the virtual circuit connection once, the sending terminal must wait 1 μsecond before sending another RVC frame. If that attempt fails, it must then wait 2 μseconds, and then 4, and so on up to 128 μseconds, after which the cycle repeats itself until the number of attempts to establish a virtual circuit has exceeded the retry count. If this happens, the sending terminal will no longer be reserved for communication only with the receiving terminal. It is important to note that the retry delays are fixed times that double with each retry and not random time periods chosen from an interval that doubles in size for each retry as in the Binary Exponential Backoff algorithm used in Ethernet.

3.5 L-Expressnet

L-Expressnet was developed for a bus network known as Campus Net (C-NET) by the Consiglio Nazionale delle Ricerche of Italy[BORG85]. It is similar to the Expressnet protocol developed at Stanford and even more similar to the BID protocol for bidirectional buses.

In L-Expressnet, the token is passed virtually rather than through the reception of an actual message or control signal. Each terminal is equipped with several timers (whose functions will be explained later) that indicate when a terminal is in possession of the token. In order for the L-Expressnet protocol to work properly, all terminals on the bus must have a unique index that reflects the terminal's spatial position on the bus. For example, the terminals may be indexed from left to right, each terminal having a higher index than the one on its left.

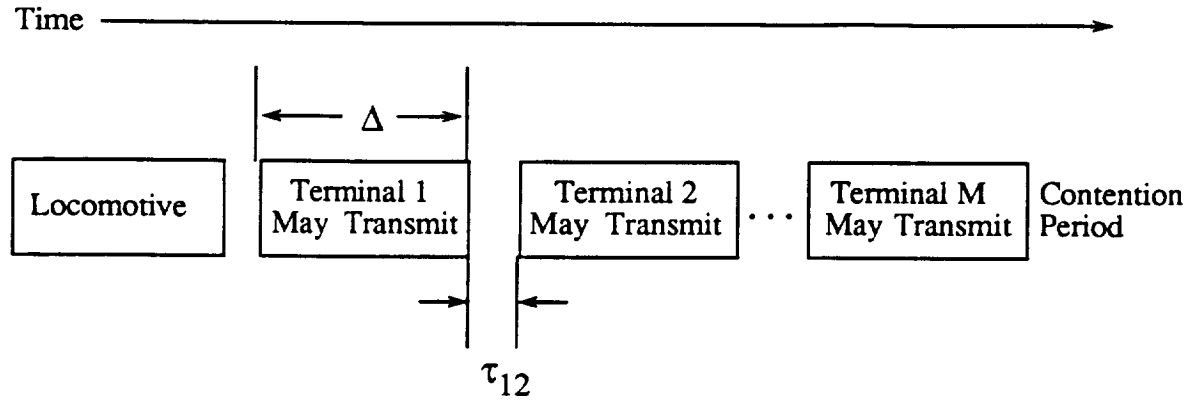
At the beginning of a cycle, or "train" in L-Expressnet terminology, all of the timers on the enabled terminals on the bus are reset and begin counting upon the detection of a signal known as the "locomotive." The locomotive need not be a string of ones and/or zeroes. It may be simply a burst of the carrier signal or the first 0 to 1 transition after the end of the previous train. Sample trains are shown in Figure 3.18.

A terminal of index i knows it is in possession of the token when a counter, CR1, reaches the value:

$$CR1(i) = (i - 1) \times \Delta$$

where Δ is a length of time greater than the reaction time, i.e. the time it takes for a terminal to detect and act upon a carrier transition on the bus. After a terminal's CR1 counter has reached the value described above there is a time period of length Δ in which it may transmit a packet. The transmission is guaranteed not to experience a collision. Although a terminal may or may not have data packets to transmit when it is in possession of the token, there will be a time of length Δ before another terminal may transmit. If a terminal does transmit when in possession of the token, the end of its transmission marks the end of the train.

After a train has traversed the bus, a new locomotive must be generated by the lowest indexed enabled terminal on the network. To know whether or not it should generate a new locomotive, each terminal is equipped with another counter,



Δ = Time in which Terminal May Transmit Guaranteed of No Collision

τ_{12} = Propagation Delay from Terminal 1 to Terminal 2

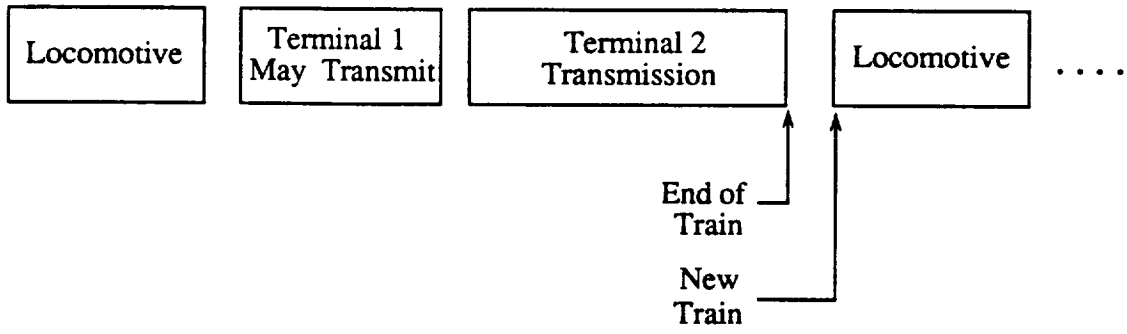


FIGURE 3.18 Sample L-Expressnet Trains

CR2, that also begins counting upon the detection of the locomotive. A terminal may generate a locomotive if its CR2 reaches a value that reflects the amount of time required for a train to traverse the network (M * Δ, where M represents the number of terminals on the bus), plus the amount of time it would have taken for a train generated by a terminal of lower index to reach it. That is, a terminal of index i may generate a locomotive if :

$$CR2(i) = M * \Delta + 2 * \tau + 2 * (i - 1) * \theta$$

where τ is the time it takes for a signal to traverse the length of the bus and θ is a time such that

$$\theta > \max \left| \tau_{ij} / (i-j) \right|$$

where τ_{ij} represents the propagation time from terminal i to terminal j.

When the network first begins operation, the CR1 and CR2 counters are at zero and will remain so until the terminal detects a locomotive as described above. However, unless the network is somehow initialized, no locomotive will ever be generated. Therefore, each terminal is equipped with a third counter, CR3, that begins counting when the terminal is first powered up but reset by the detection of the locomotive. When a terminal's CR3 counter reaches a value given by

$$CR3 = M * \Delta + 4 * \tau + 2 * (M - 1) * \theta$$

it knows that the network is not initialized and may generate a locomotive. The first locomotive is known as the "pilot."

3.6 PERFORMANCE ANALYSIS OF THE ETHERNET, HYPERchannel AND ProNET-10 PROTOCOLS.

In order to determine the applicability of a given protocol to a system, the protocol's performance must be studied. One of the most important performance factors is average delay. In the following sections, offered load versus delay is plotted for several protocols. The data points for these plots were obtained through computer simulation of the protocols and from various studies of protocol performance[HEAT86/FRAN84/LIU82/OKAD84].

The performance of CSMA/CD and token ring protocols has been compared often in the literature[LIU82/ OKAD84]. The basic results are that for lightly loaded

systems the performance of CSMA/CD is better than token ring in terms of average delay. This is because a station may access the channel any time it is free in the CSMA/CD system but must wait while the token is passed through stations that are not ready to transmit in the token passing ring system. As the system becomes more heavily loaded, collisions occur more often in the CSMA/CD system and time is spent in the backoff period. For heavily loaded systems the token passing protocol is recommended because the token passing overhead is relatively small if a majority of the stations have traffic.

Another aspect of performance is a guarantee of data delivery. A guarantee of delivery can be made in a CSMA/CD system with acknowledgment, but the acknowledgment packet also requires channel resources. In the token passing system, a flag is usually located in the packet that the receiving station uses to inform the sending station that the packet was received with no detected errors. Another advantage of the token ring system is guaranteed access because a CSMA/CD system cannot guarantee access. This makes the token system more applicable to real time systems that require guaranteed access and delivery.

The HYPERchannel protocol is advantageous for some applications because of its prioritized access scheme. The service of low priority stations suffers for the improvement in high priority service.

MIL-STD-1553B is useful in systems with one central controller and tight requirements on access and delivery. It is not well suited to systems that have more than one source that generates traffic in a Poisson manner. This is a serious problem in avionics systems where smart distributed systems are becoming prevalent. The use of a command/response system defeats one of the advantages of distributed intel-

ligence in that the system must wait for the main controller to allow an intelligent subsystem access to the bus which could not occur until the next time the controller was scheduled to correspond with that station. The subsystem cannot ask for additional sensor information or main memory access until the main controller allows it access to the bus.

3.6.1 PERFORMANCE ANALYSIS OF ETHERNET

The delay vs. offered load and packets lost vs. offered load plots shown in Figures 3.19 and 3.20 were taken from the paper "A Simplified Discrete Event Simulation Model for an IEEE 802.3 Local Area Network" by Sharon K. Heatley of the National Bureau of Standards[HEAT86]. These are the results of a computer simulation of the IEEE 802.3 protocol standard which has the same timing features as Ethernet. A typical simulation timeline is shown in Figure 3.21. The protocol was modeled using the following rules:

1. The arrival of packets at each terminal is a Poisson distributed random process.
2. The propagation delay between any two stations is constant. This would be the case for an Ethernet network on which the terminals are equally spaced along the propagation medium.
3. After a collision, all terminals involved go into a back-off period, the length of which is determined by the binary exponential backoff algorithm.

and the following parameters:

1. Slot time=51.2 μseconds
2. Interframe gap (I)=9.6 μseconds
3. Jam size (J)=3.2 μseconds
4. Maximum propagation delay=25.6 μseconds

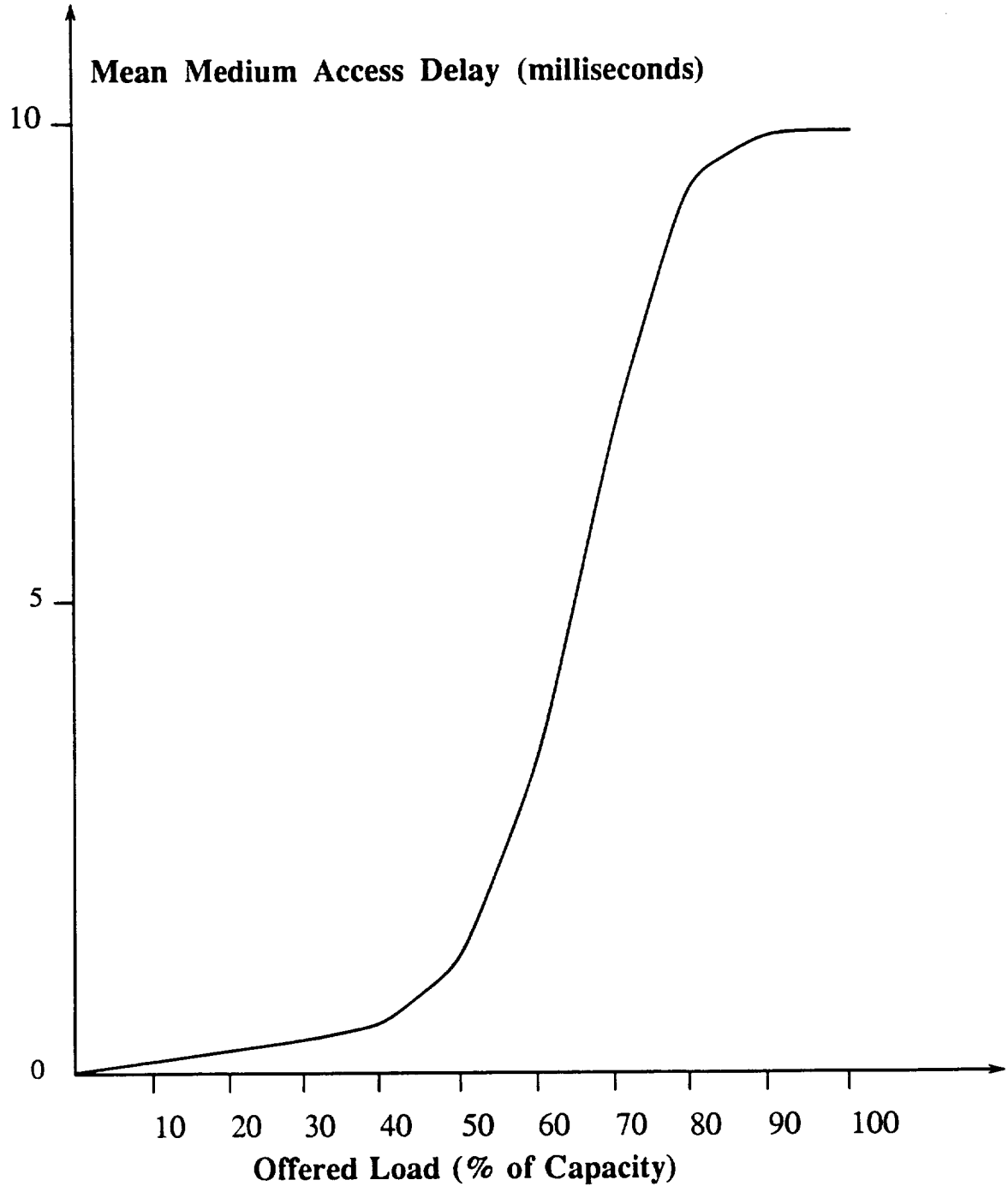


FIGURE 3.19 Mean Medium Access Frame Delay vs. Offered Load for Ethernet [HEAT86]

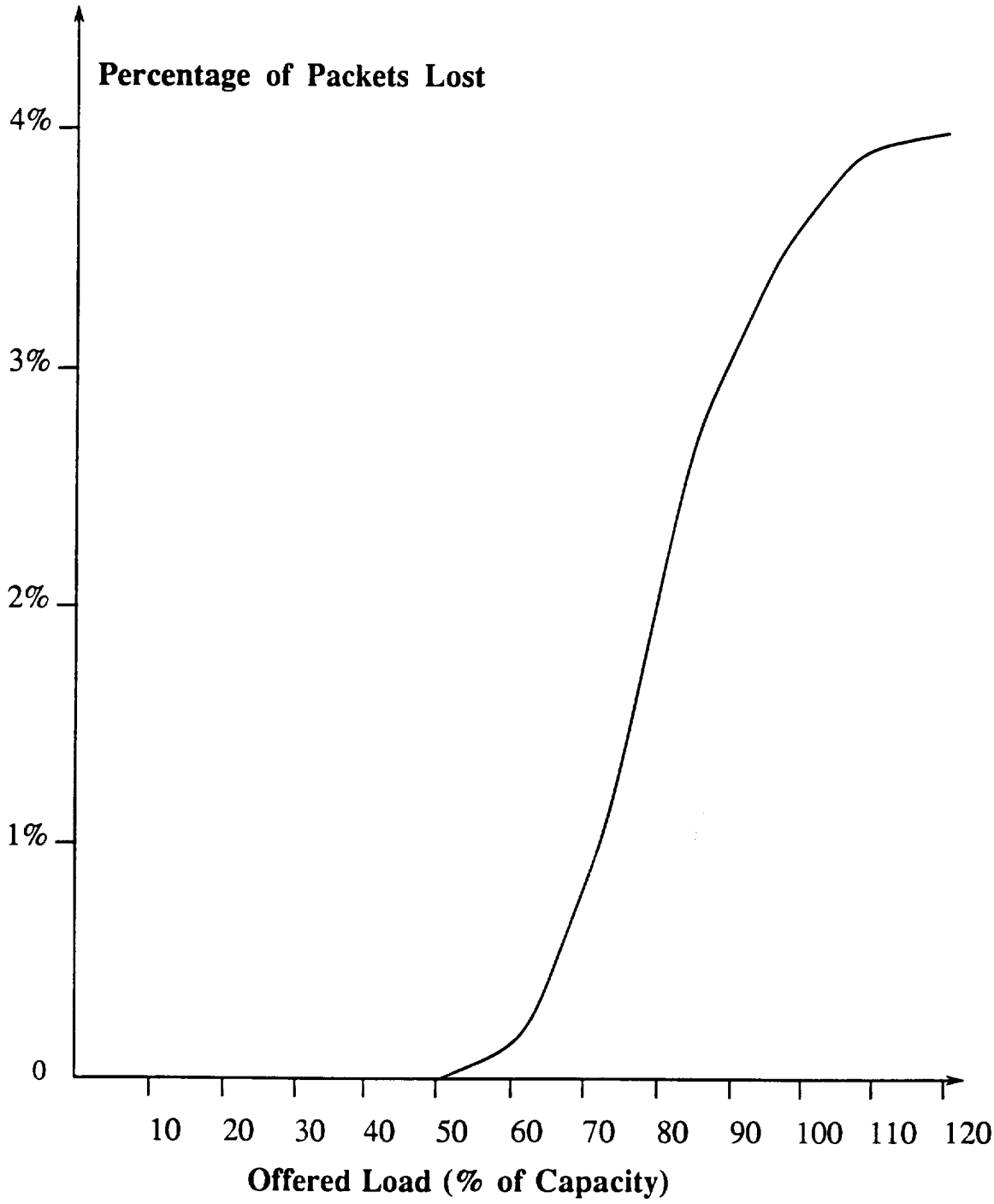


FIGURE 3.20 Percentage of Lost Packets vs. Offered Load for Ethernet [HEAT86]

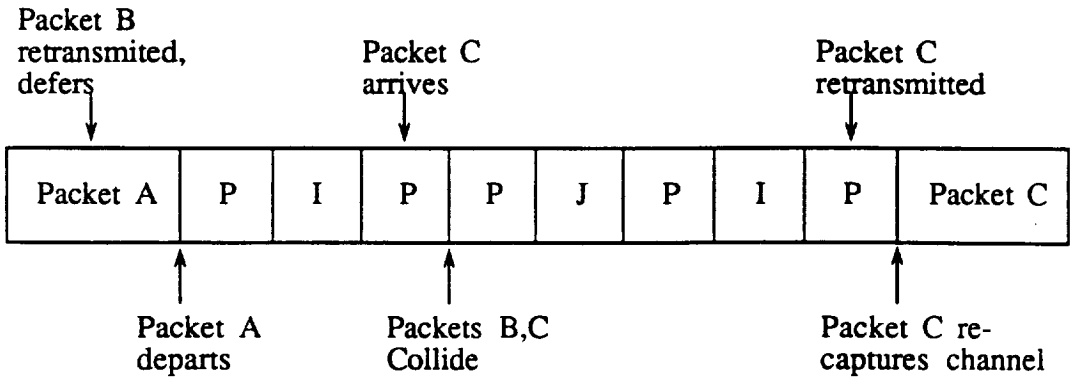


FIGURE 3.21 Typical Timeline for Simulation Model [HEAT86]

5. 20% of packets are 1024 bytes in length and 80% of packets are 64 bytes in length.

3.6.2 PERFORMANCE ANALYSIS OF HYPERchannel

This section gives the results of a HYPERchannel simulation presented in the paper "Measurement and Analysis of HYPERchannel Networks" by William R. Franta and John R. Heath. A detailed description of the HYPERchannel protocol can be found in section 3.4.

The HYPERchannel network simulated was composed of six terminals connected to a 1000 foot bus. Three of the terminals were designated "senders" and served only as data sources. The remaining three terminals were designated "receivers" and served only to colllect data. The "data" was generated by a thirty-two bit random number generator. Each node was provided with a separate "seed" for the random number generator in order to avoid the repeated transmission of identical frames.

Several simulations of over 110,000 frame sequence transfers yielded results

within 2% of each other. Figures 3.22 and 3.23 show the average delay normalized to "transmission period units" plotted against the percentage of 50 Mbits/second of the offered load. Figures 3.24 and 3.25 show the throughput, i.e., the trunk utilization versus the offered load.

It was also observed that the effect of the wait flip-flop was not what the designers of HYPERchannel expected. Instead of preventing the higher priorities from "hogging" the channel, it has reversed the priorities with every frame sequence[FRAN84].

3.6.3 COMPARATIVE RESULTS FROM ANALYTIC AND SIMULATION STUDIES OF CSMA/CD AND RING PROTOCOLS

1- Comparison between the delay characteristics of the token ring and CSMA/CD protocols [LIU82]

1- Conditions under which the comparison was conducted:

- Normalized transmission media = 0.005
- Token ring packets are removed by the source.

2- Results are shown in Figure 3.26.

3- The following can be concluded from the results:

- At light loads, the token ring suffers greater delay than CSMA/CD.
- At heavy loads the token ring protocol suffers less delay than CSMA/CD.
- At heavy loads the token ring's delay appears to be stable.

2- Comparison between the delay characteristics of the token ring and CSMA/CD protocols [OKAD84]

1- Conditions under which the comparison was made:

- Channel Capacity = 5 Mbps.,
- Number of nodes used = 50.

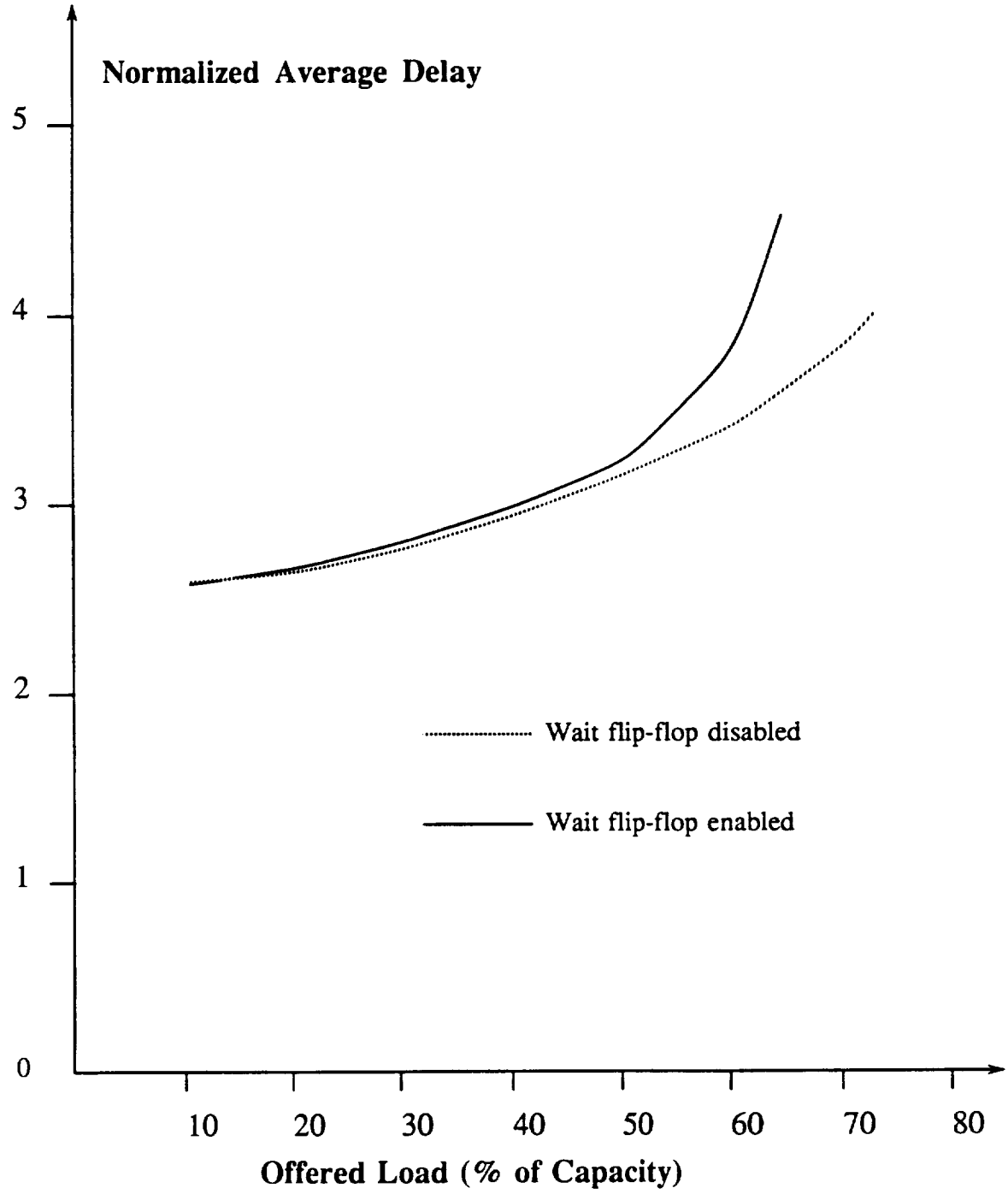


FIGURE 3.22 Average Delay for 64-Byte Message-only Sequences vs. Offered Load for HYPERchannel

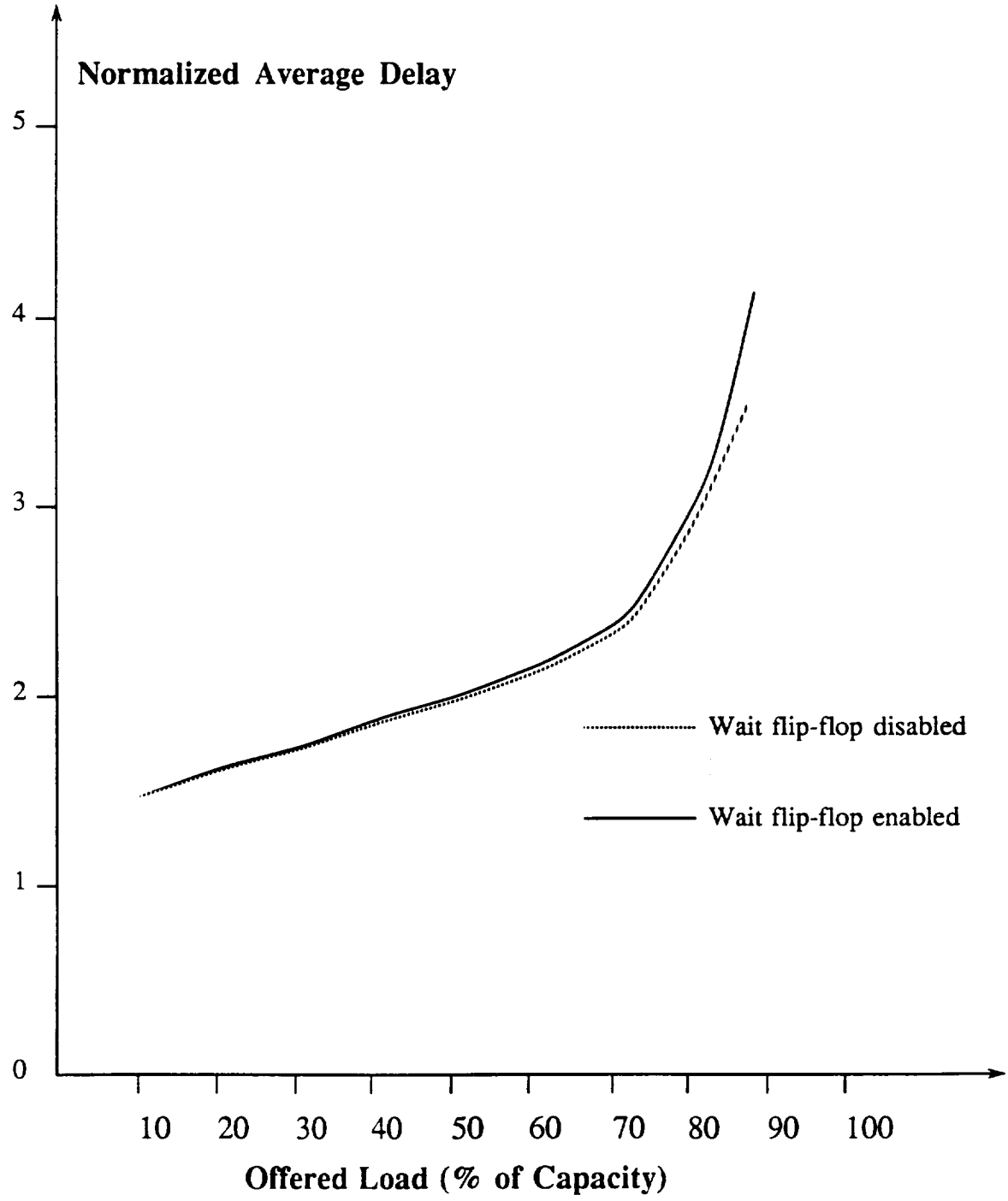


FIGURE 3.23 Mean Medium Access Frame Delay vs. Offered Load for HYPERchannel

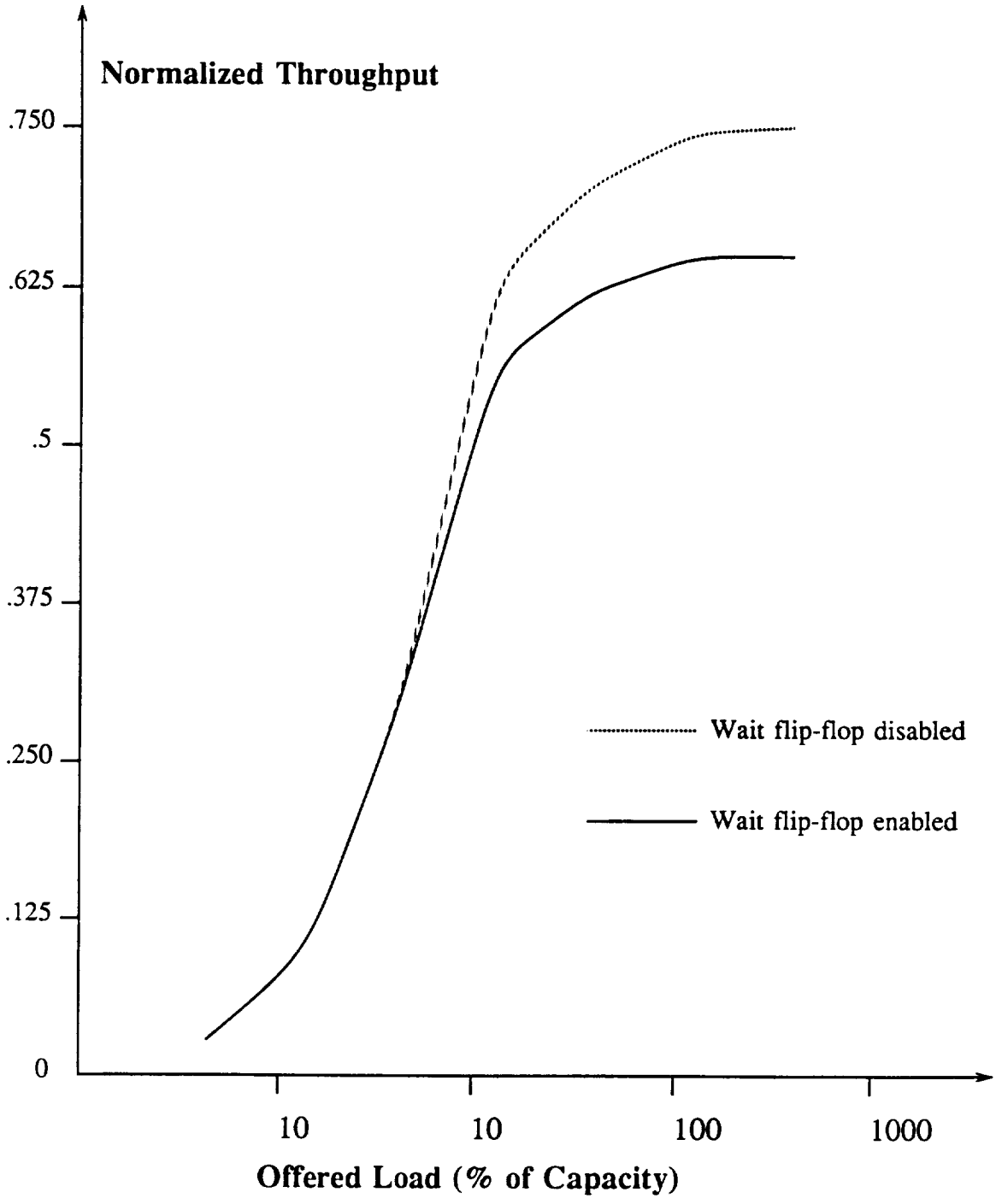


FIGURE 3.24 64-Byte Message-Only Sequence Throughput vs. Offered Load for HYPERchannel

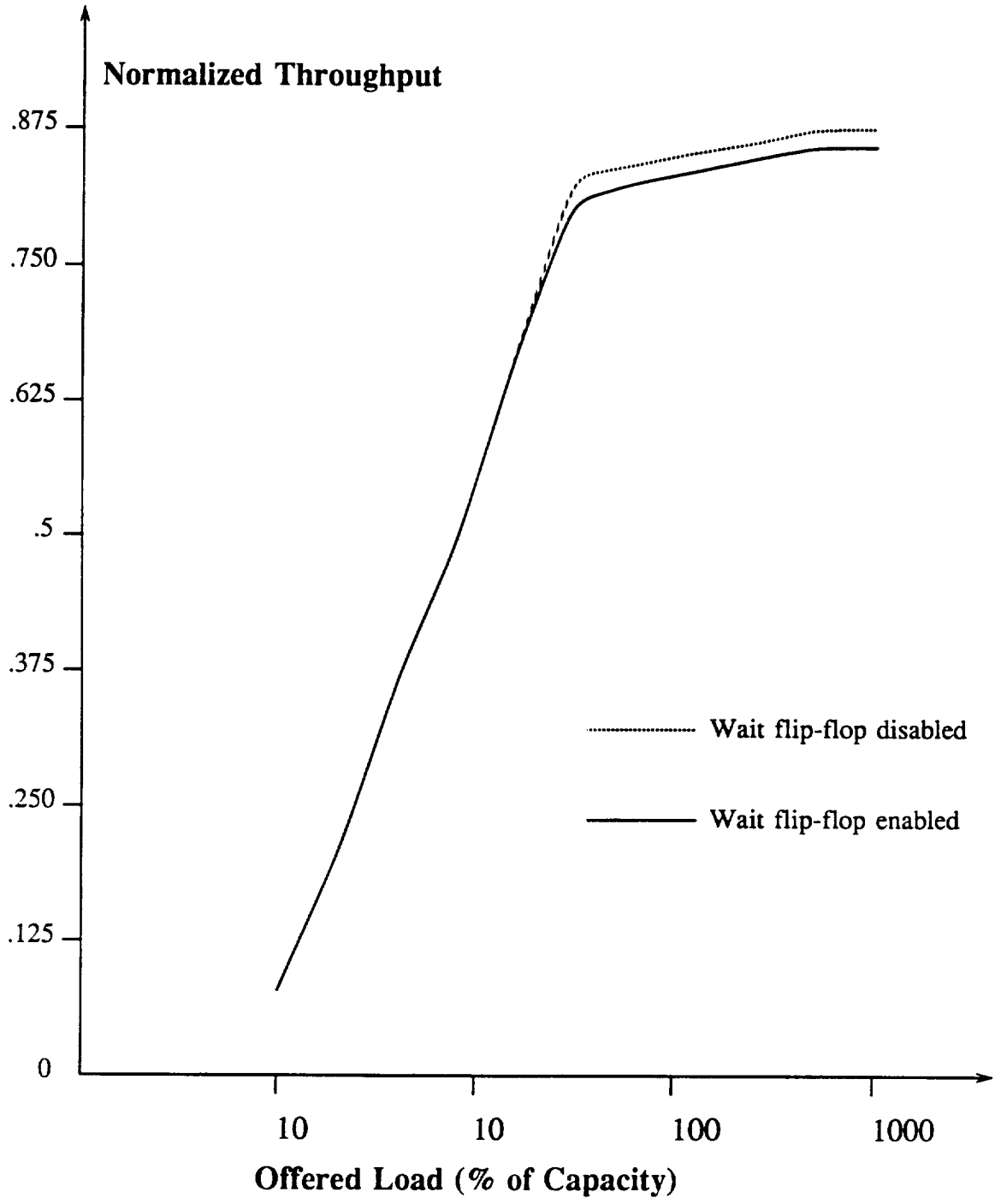


FIGURE 3.25 64-Byte Message Proper with 4 KByte Data Sequences Throughput vs. Offered Load for HYPERchannel

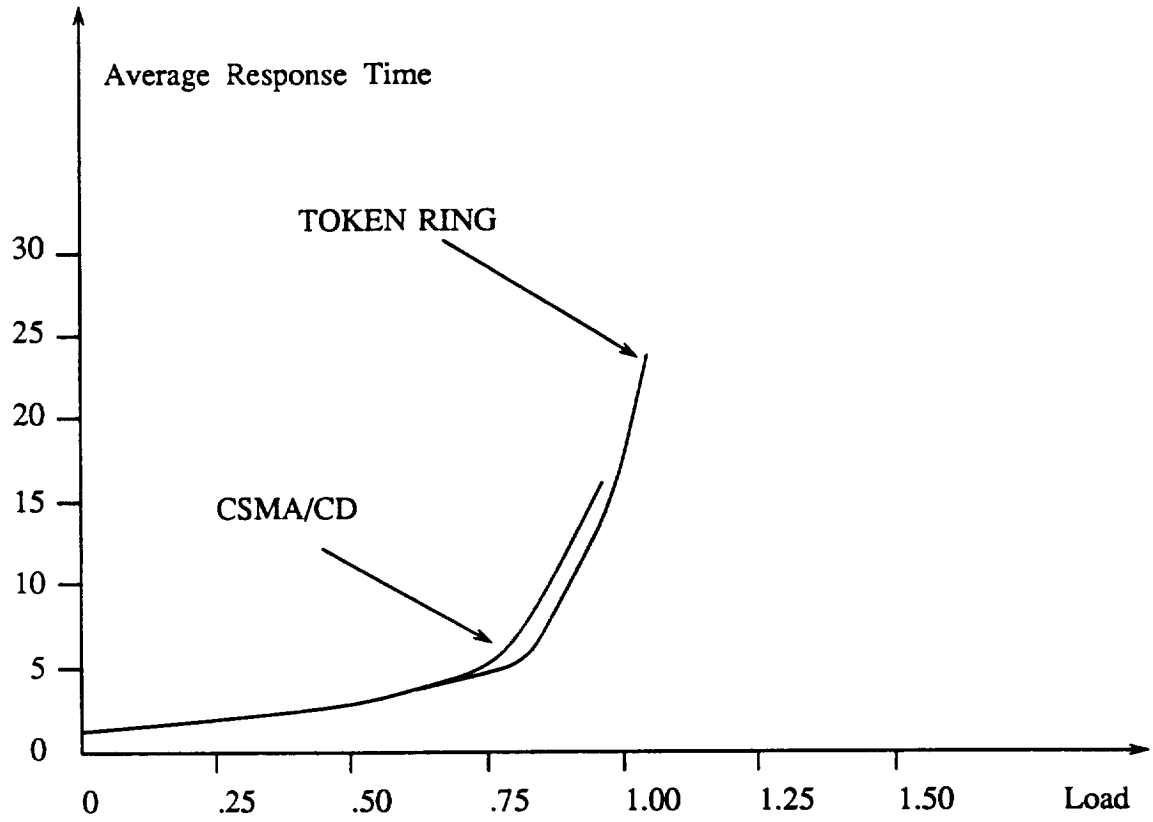


FIGURE 3.26 Average Response Time Comparison

- Maximum distance covered = 1.0 Km.

- Packet length = 1000 bits.

- Repeat delay at each node = 8 bits.

- Token header length = 24 bits.

- Maximum length of contention to control = 7. (CSMA/CD binary exponential back-off)

2- Results are shown in Figure 3.27.

3- The following can be concluded from the results:

- At light throughput values, CSMA/ CD protocol experiences less delay than the token ring protocol.

- At heavy throughput values, the token ring protocol experiences less delay than the CSMA/CD protocol.

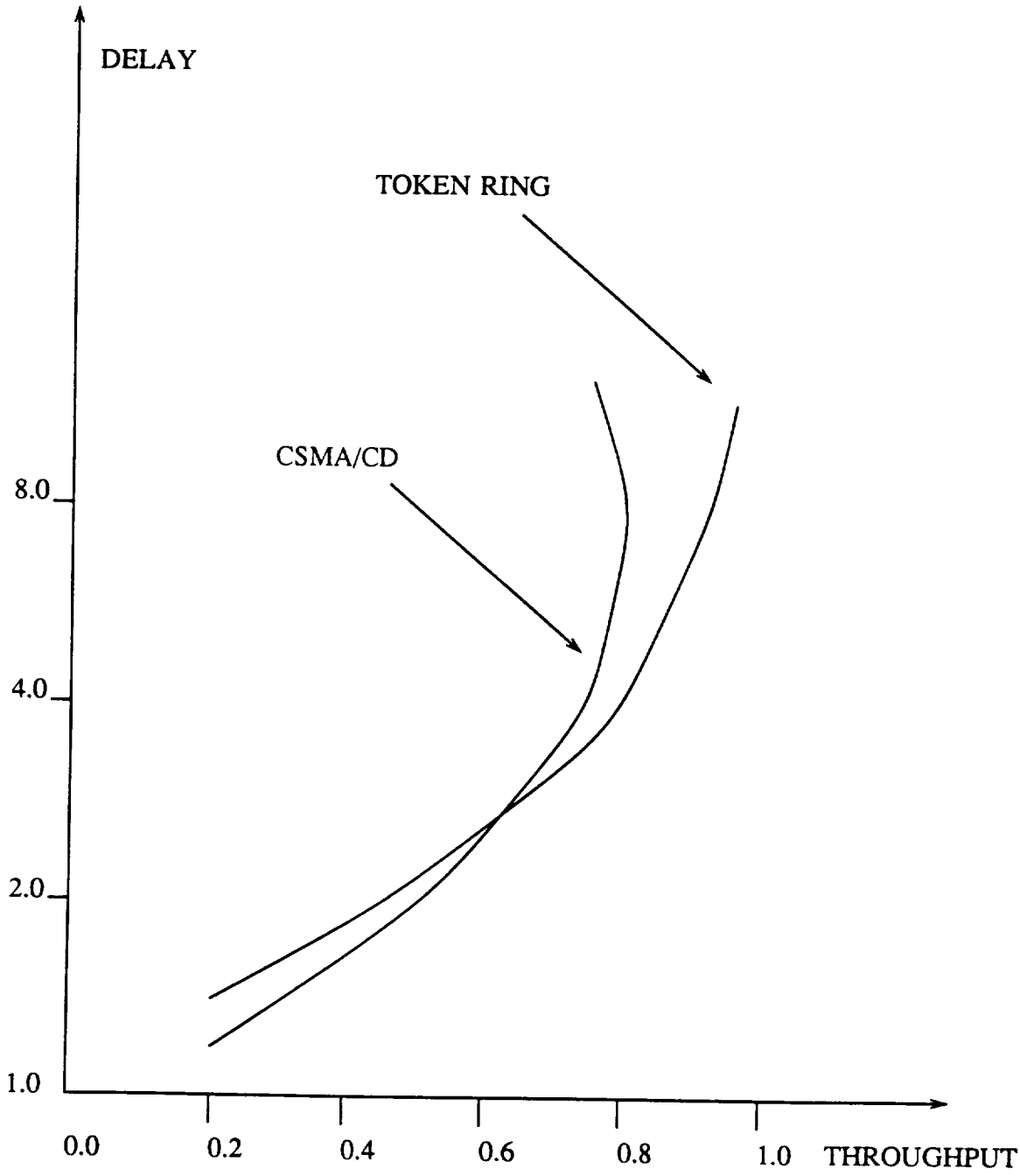


FIGURE 3.27 Throughput-Delay Characteristics Comparison

4 MODIFIED FREE ACCESS PROTOCOL

4.1 THE COMMUNICATIONS SYSTEM OF INTEREST

The communications systems of special interest to this study are those that have a majority of Poisson distributed traffic with some time critical or scheduled periodic traffic as may be found in data collection systems and other real time applications. These systems are not well served by CSMA/CD/BUS, TOKEN/RING, or COMMAND/RESPONSE/BUS protocols. CSMA/CD protocols are unable to guarantee data latency because any packet may suffer multiple collisions even on a lightly loaded system. The token ring protocols guarantee data latency at the cost of poor performance for light Poisson loads. The command/response protocols provide good performance for the scheduled loads but are not well suited for Poisson traffic. The communication needs of systems that have a majority of Poisson distributed communications with some time critical or scheduled periodic traffic could be served by a communications protocol that provided a period of either scheduled or command/response access followed by a period of free access.

4.2 MODIFIED FREE ACCESS PROTOCOL

The MFA protocol resides in level one, the physical layer, of the seven layer model. The modified free access (MFA) protocol is a modified implementation of the Ethernet level 1 protocol. It is designed to serve systems with both scheduled and Poisson traffic. For the MFA protocol, time is divided into communications cycles. The cycle time is variable and should be set so that the access requirements of the scheduled traffic may be satisfied. Each cycle is divided into two sections. In the first section, stations have exclusive access to the bus and this is followed by a section of

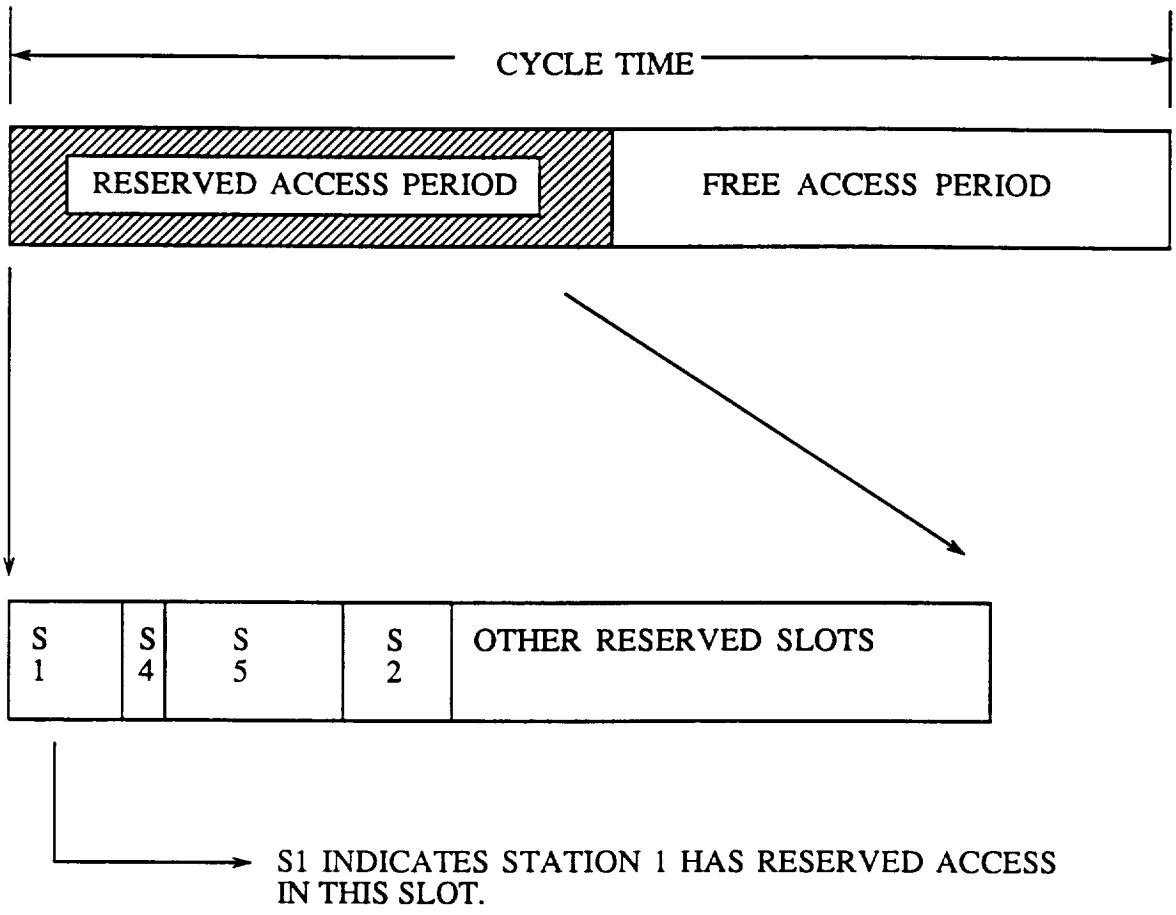
Ethernet access. A time line for MFA is shown in Figure 4.1. Each station's access may be described by the algorithm illustrated in the flowchart shown in Figure 4.2.

The advantage of this protocol over the Ethernet and HYPERchannel protocols is that stations with real time or time critical packets can be assigned slots in the reserved access period. This guarantees access time for these stations. This feature makes this protocol attractive for systems with even very small amounts of high priority scheduled traffic. The MFA protocol also has an advantage over token ring protocols in that stations may be given a guaranteed access time while the network's performance for light Poisson traffic is comparable to CSMA/CD. This performance figure assumes that the reserved access slots are utilized by the scheduled traffic 100% of the time. The MFA protocol degenerates to the Ethernet protocol in systems were there is no scheduled traffic.

For ease of implementation the protocol was designed so that standard Ethernet terminals may be used for channel access. This is valuable for hardware availability and economic reasons. This restriction should not limit the protocol's applications but is an advantage because the protocol may be used in cost sensitive applications.

4.3 IMPLEMENTATION

There are several methods by which modifications of the standard Ethernet terminal required to implement the MFA protocol may be accomplished. The two modifications that are most obvious to implement are hardware modifications to the Ethernet terminal hardware or modifications to the host's network interface software. The hardware modification is preferable because it leaves the cpu cycles that are required for the execution of additional software free for the execution of the sta-



NOTE: RESERVED TIME SLOTS MUST BE A MULTIPLE OF 1 MICROSECOND DUE TO TIMER CONSTRAINTS AND GREATER THAN 57 MICROSECONDS, THE MINIMUM ETHERNET PACKET LENGTH.

FIGURE 4.1 MFA Cycle Definition

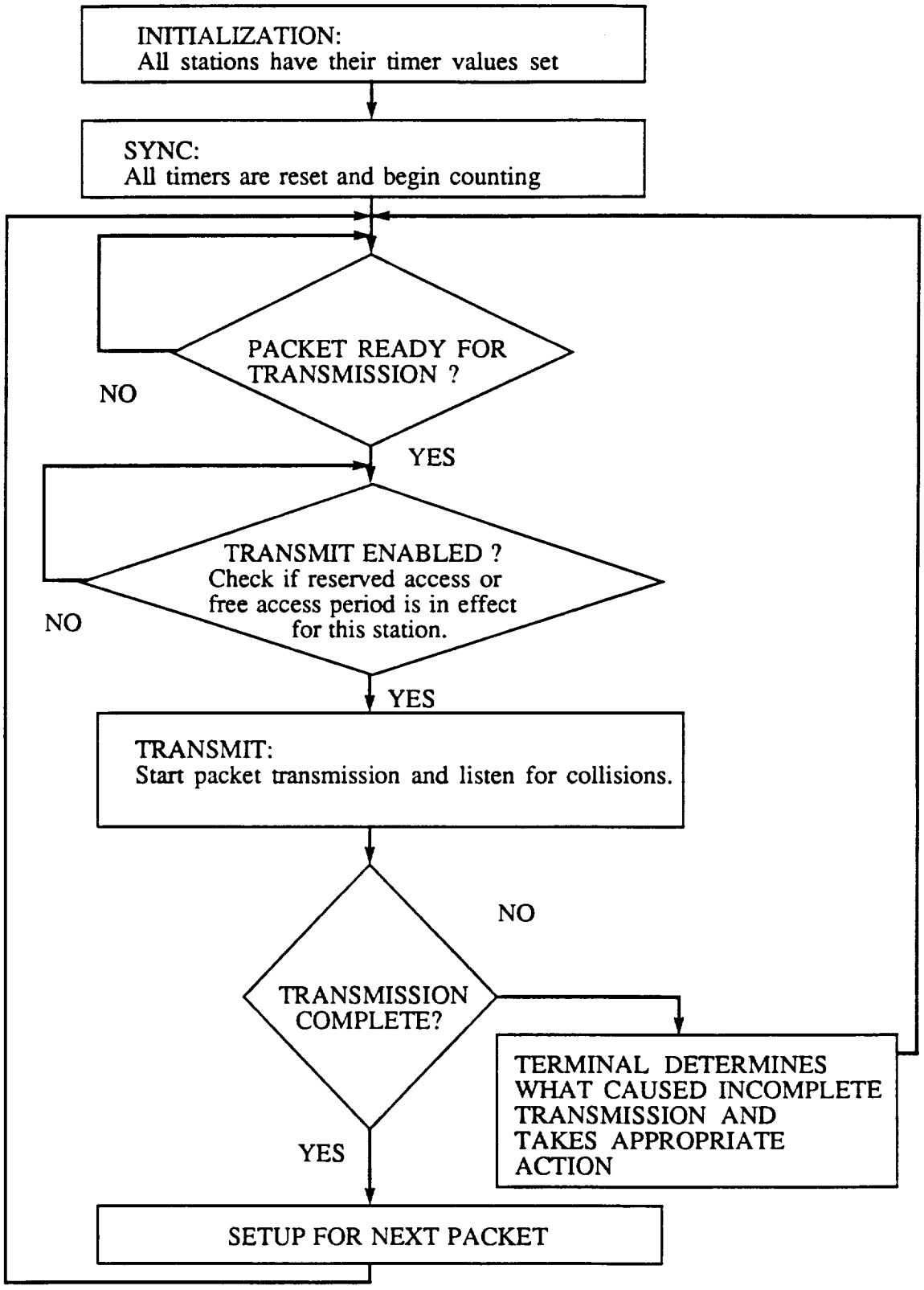


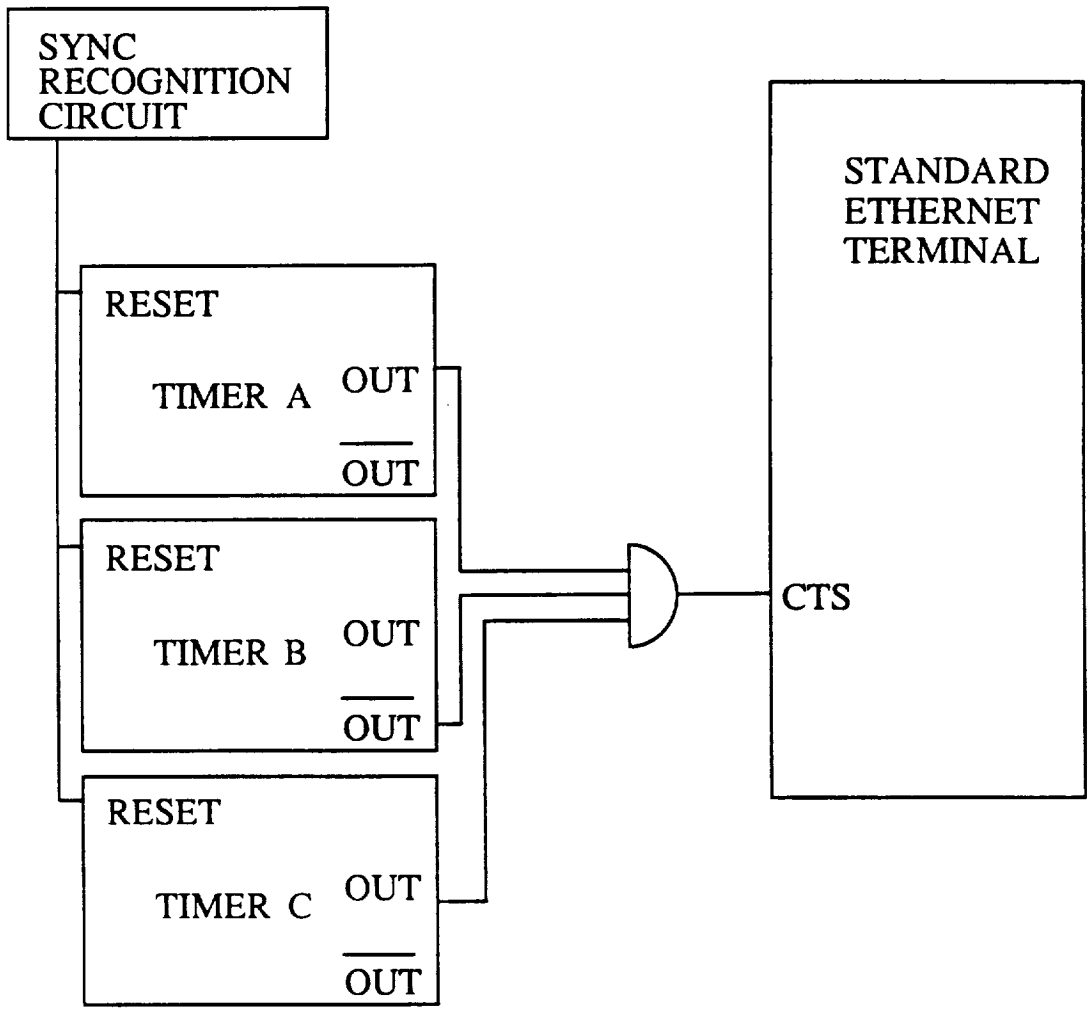
FIGURE 4.2 Transmission Flowchart for MFA Stations

tion's application software. The hardware modification is also preferable in that a system with diverse processors and standard communications interfaces could easily be implemented. The use of a software solution in the diverse processor system would require a version of the network software to be written for each of the different machines. The programming time and debug time for these modifications would make implementation of the system unattractive.

The use of external hardware for modification of the carrier sense line to control Ethernet access has been shown effective in the L-Expressnet project [BORG85]. The major difference between MFA and L-Expressnet is in the modification of the Ethernet standard. In the L-Expressnet project the protocol implemented by the Ethernet protocol integrated circuits was in essence a virtual token passing bus. The first access method of the MFA protocol may be viewed as a virtual token passing protocol similar to the L-Expressnet and HYPERchannel.

A simple hardware approach is to utilize the clear to send (CTS) signal provided on typical Ethernet interface integrated circuits. The CTS line could be driven by either a set of timers as shown in Figure 4.3 or a small dedicated microprocessor system as shown in Figure 4.4. Both of these methods accomplish the same task with the timer addition being the simplest and the microprocessor addition being the most versatile. The microprocessor system has an advantage in that the microprocessor could be used to recognize the sync pattern and other system timing messages. This service must be provided to the timer controlled system by the host computer or additional hardware.

The timer modification could be easily accomplished in two ways. The first way would require the use of three timers and a system sync signal which would be



NOTE: TIMER VALUES ARE FIXED

HOST COMPUTER OR EXTERNAL
HARDWARE MUST RECOGNIZE SYNC
AND GENERATE CLOCK RESET

FIGURE 4.3 Timer Configuration to Drive CTS

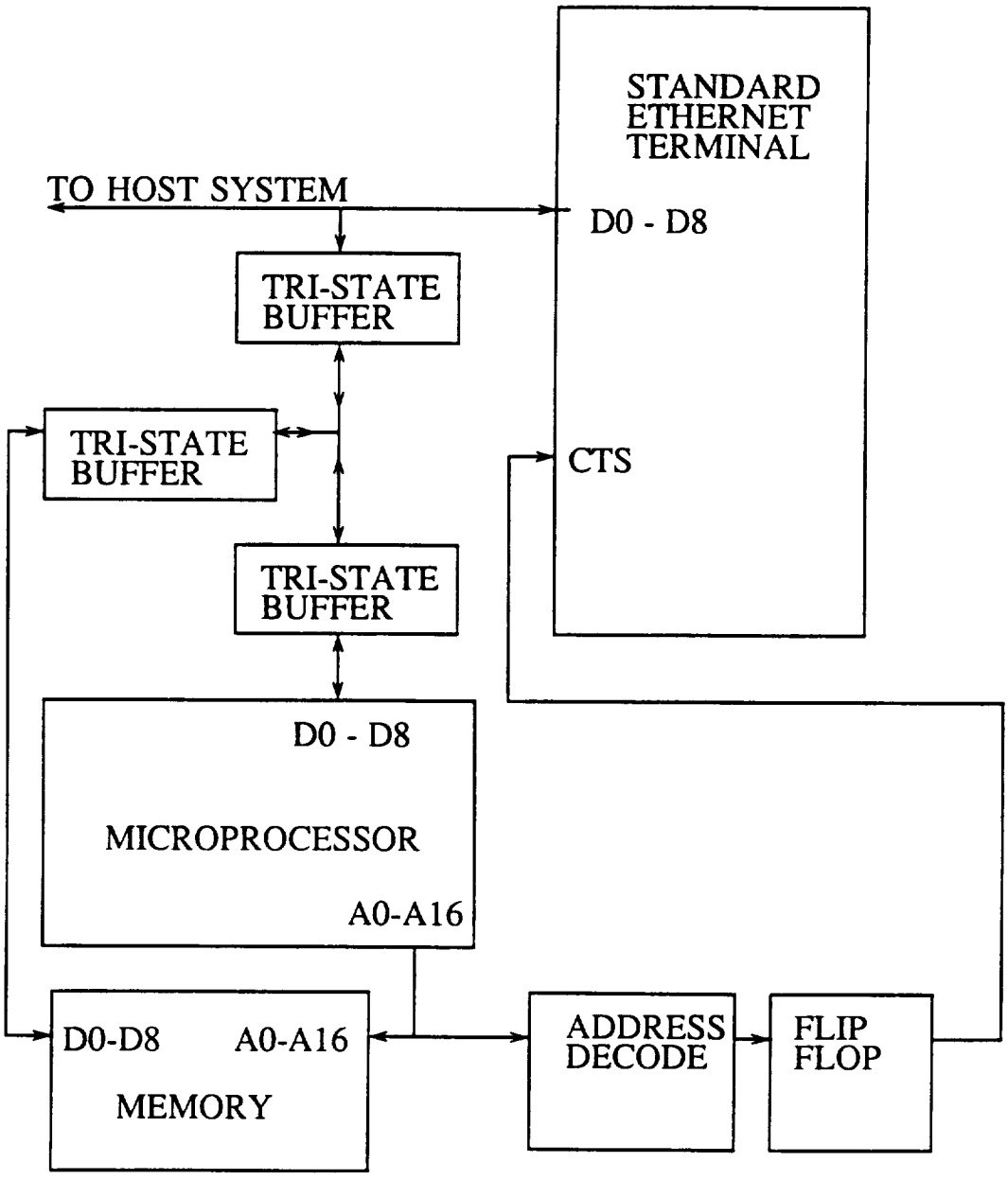


FIGURE 4.4 Microprocessor Configuration to Drive CTS

required at the start of each cycle to reset all the timers as shown in Figure 4.5. Another method would be to use four timers with timer Td set at the cycle rate as shown in Figure 4.6. Td would be used to reset all timers at the end of each cycle. The second method has the advantage of taking less of the available bandwidth for sync purposes at the cost of requiring more stringent requirements on the clocks. This is somewhat mitigated by the Ethernet protocols use of carrier sense which will allow leeway in timer accuracy.

Another method that would possibly reduce the integrated circuit count of the modification would be to provide a method to load the timer values shown in Figure 4.7 into one timer. The outputs of the timers would be combined through high speed digital logic to drive the clear to send or carrier sense inputs of Ethernet terminal integrated circuits. The use of the clear to send line allows the collision detect and receive circuitry to operate without interference while the transmit section is only enabled when the timers indicate a reserved period for the station or the free access period at the end of each cycle.

The timers will be driven by the ten megahertz clock provided by all Ethernet stations. This will give one hundred nanosecond resolution to the timing periods which will usually not be needed. The ten megahertz clock should first be run through a divide by ten counter to provide 1 microsecond resolution or a divide by 100 counter to provide 10 microsecond resolution. This reduces the values that the protocol timers must count and is more than adequate resolution since there is a minimum of 9.6 microseconds quiet period at the end of each transmission and the minimum packet transmission time is 57.6 microseconds. Another factor allowing low resolution timers is carrier sense access requirements of Ethernet. This allows the next station

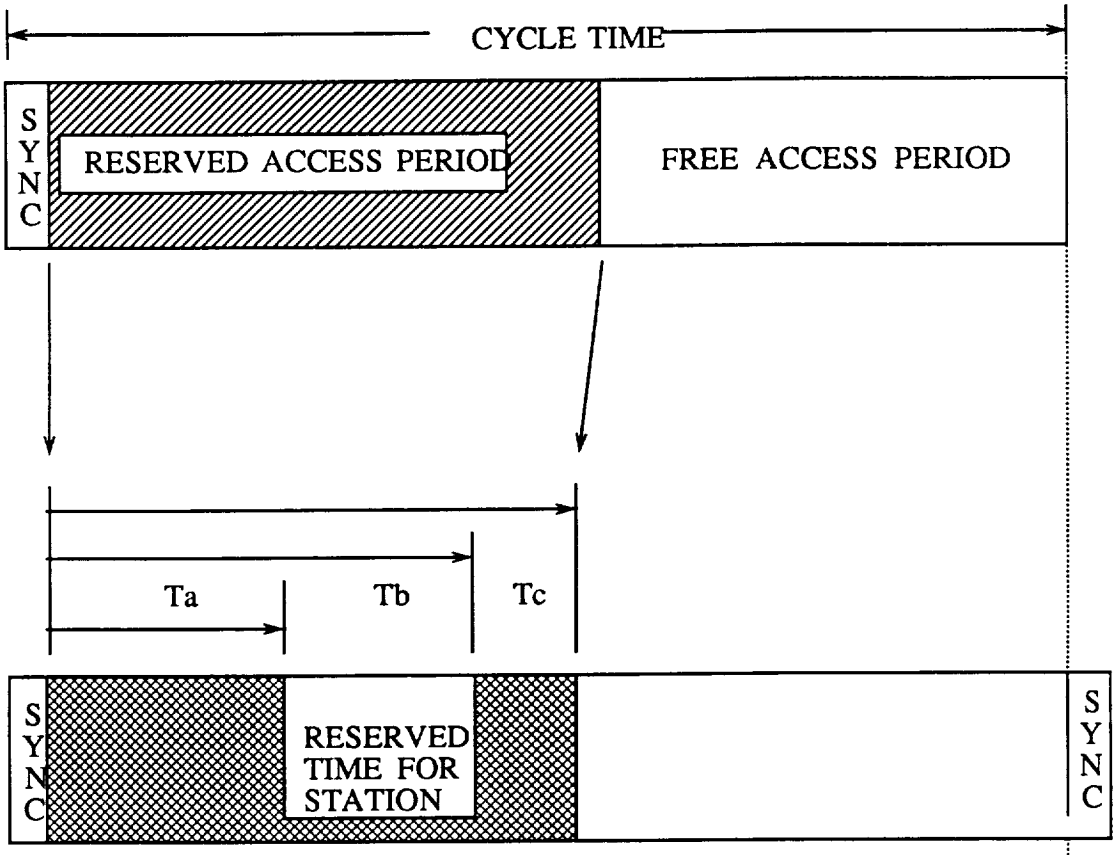
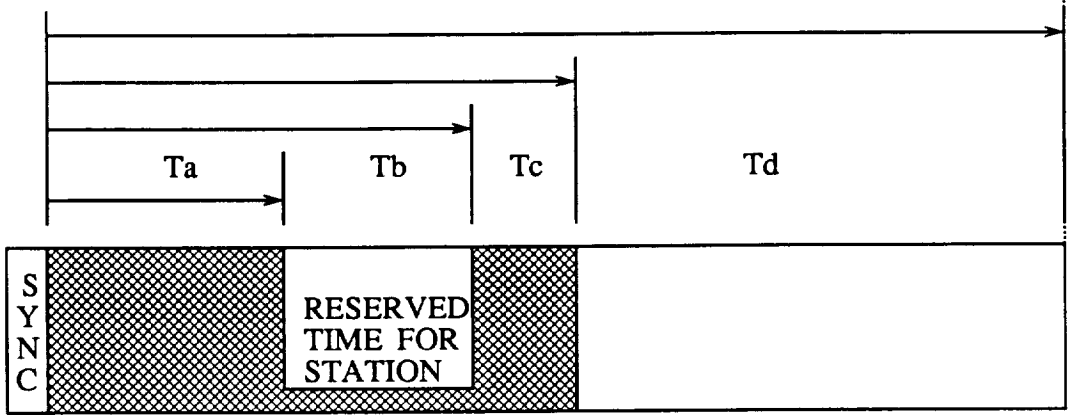


FIGURE 4.5 MFA Implementation with Three Timers and Sync Every Cycle



NOTE: SYNC IS SENT ONLY EVERY N CYCLES WITH N DETERMINED BY THE SYSTEM TIMING ACCURACY REQUIREMENTS AND CLOCK STABILITY

FIGURE 4.6 MFA Implementation with Four Timers and Sync Sent Every N Cycles

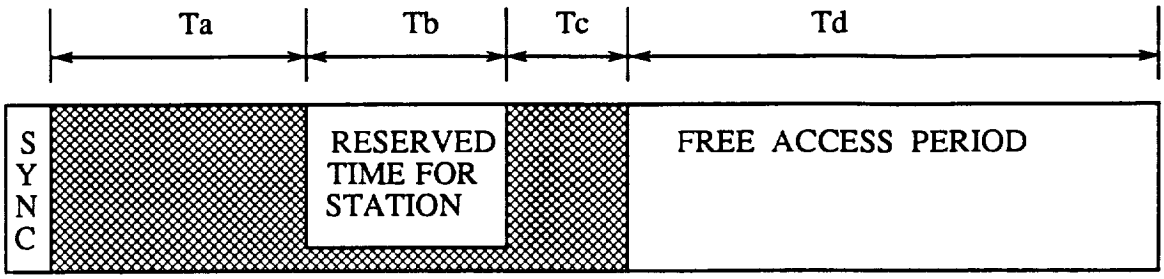


FIGURE 4.7 Alternate Timer Values

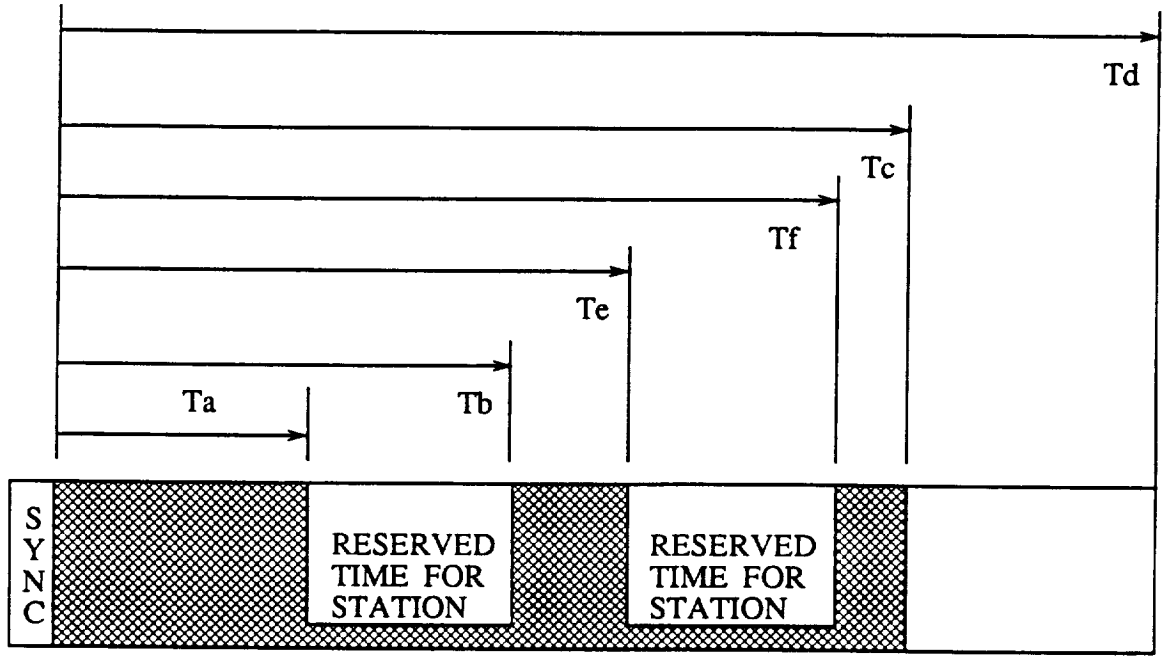
to transmit to become ready anytime after the current tranmitter's signal reaches this station. The timers and logic can be built with standard integrated circuits.

The microprocessor driven version of this protocol has the microprocessor serving in place of the timer section as described above. This would be useful if a station is to have several reserved slots in each reserved access period. However, most applications will allow the station to send consecutive packets in a single reserved slot as shown in Figure 4.8.

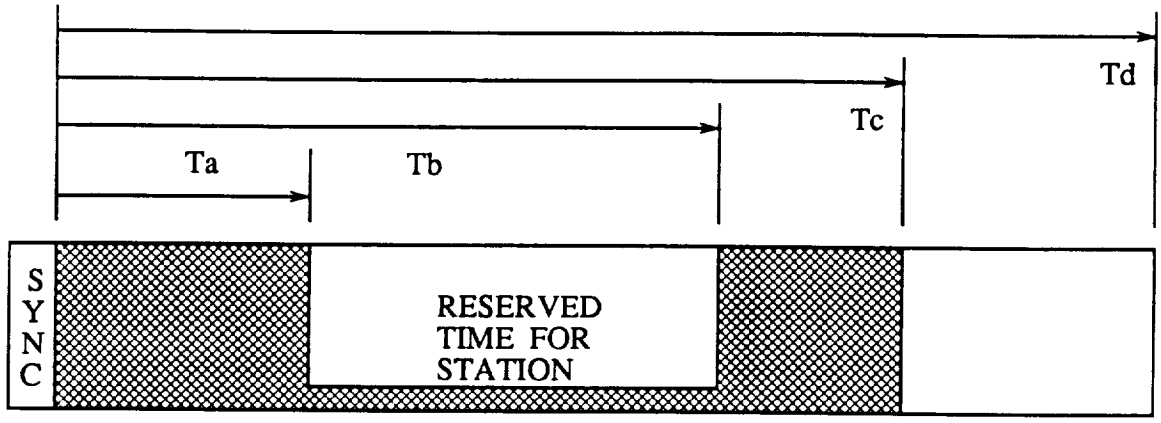
A mixture of these timer and microprocessor driven implementations could be used in the same system to satisfy the needs of various stations. The timer driven systems would be used for terminals whose packets could be grouped while the microprocessor driven systems would be used for terminals that required several slots in each reserved period.

4.3.1 IMPLEMENTATION DETAILS

A circuit diagram of a typical Ethernet terminal implementation is shown in Figure 4.9. The CTS line is used to enable the transmitter section of the terminal. The unit will behave as a standard Ethernet terminal for receiving and collision detec-



MULTIPLE RESERVED SLOTS WITHIN ONE PERIOD REQUIRES EITHER ADDITIONAL TIMERS OR A MICROPROCESSOR



A STATION MAY SEND MULTIPLE PACKETS DURING ONE RESERVED SLOT. THIS IMPLEMENTATION IS PREFERABLE AS LONG AS IT MEETS THE SYSTEMS COMMUNICATIONS REQUIREMENTS

FIGURE 4.8 Slot Assignments

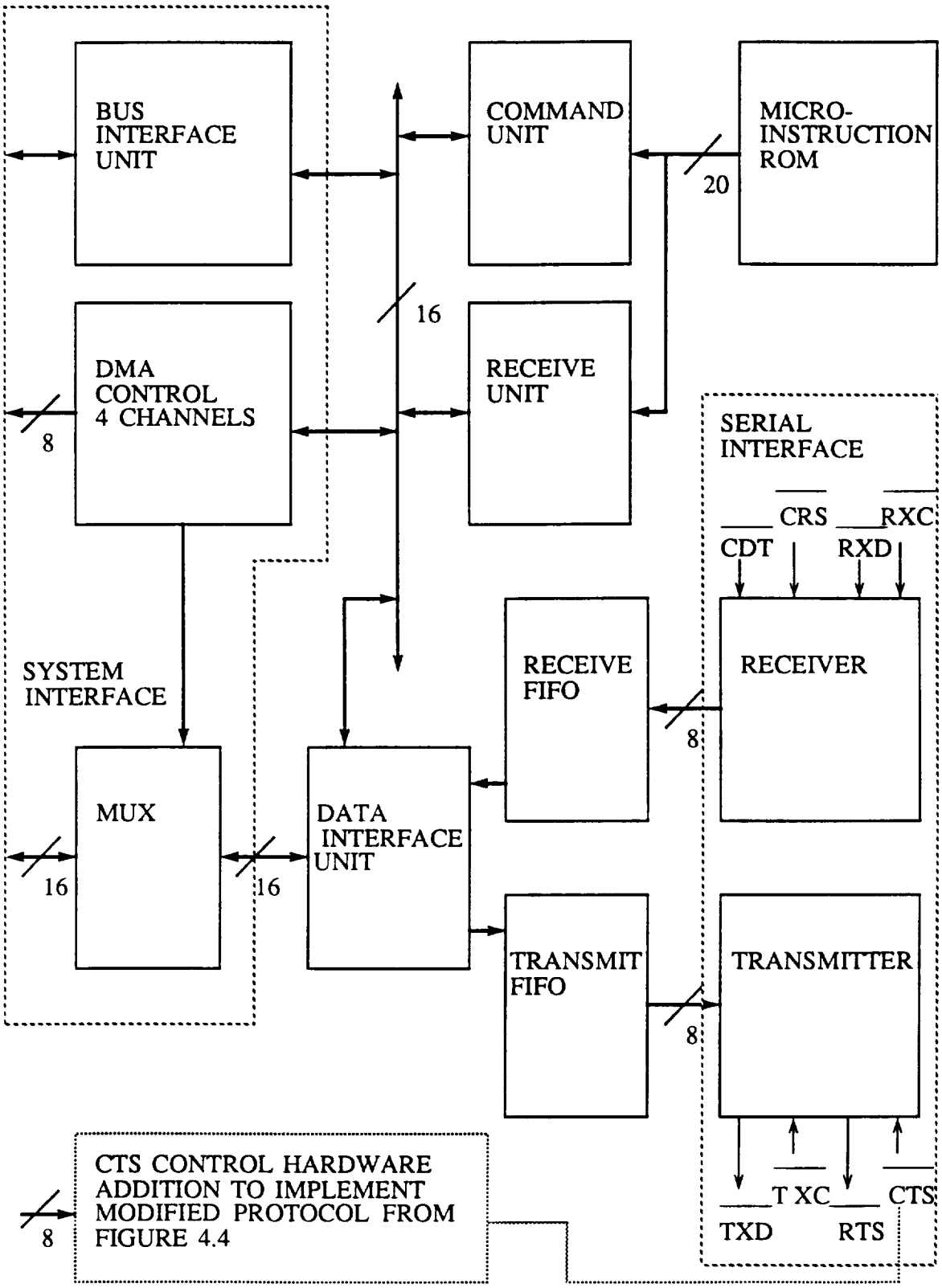


FIGURE 4.9 MFA Terminal Implementation

tion. The modifications will, however, affect the transmitter flow chart as shown in Figure 4.10 and, therefore, their effects must be understood so that appropriate action may be taken.

The terminal will behave as any Ethernet terminal if the CTS line is held high(enabled). The terminal is only affected by the CTS line if it is transmitting. If a station has a packet to transmit, it will defer the transmission until all Ethernet transmission requirements are met and the CTS line is true. If the station is transmitting and the CTS line goes low, then the station will cease transmission. This event will be recorded in the transmitter status field of the transmit command block. This event will also change the state of the command unit from active to suspended. This will result in the terminal updating the system control block status word. The status word is modified to indicate that the command unit is inactive and to indicate that the command unit left the active state. This will also cause the INT interrupt request line to go high, indicating a request for service from the host. The host must then read the system control block and determine that the command unit has gone inactive, it must then determine if the transmission was completed. If the transmission was not completed, then the host must determine why the unit failed to complete the transmission. This can be caused by exceeding the number of retries or the loss of CTS before the completion of any transmission attempt. In the case of the loss of CTS, the packet should be immediately rescheduled so that its transmission may begin at the end of the reserved access period.

Initialization and power failure recovery would be accomplished by all stations assuming sync has just occurred. This will allow only a few stations access to the

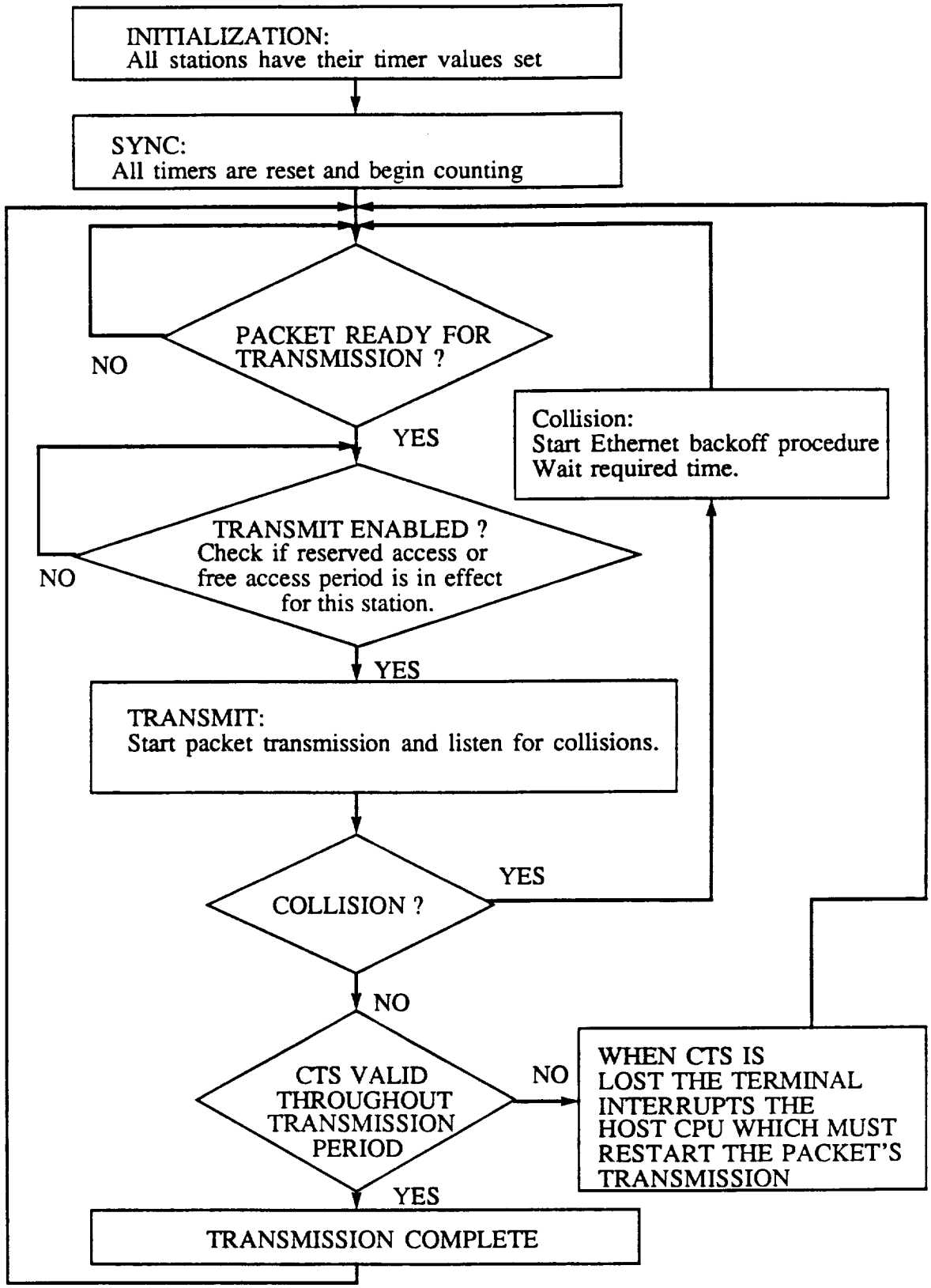


FIGURE 4.10 Protocol Flowchart for MFA Stations

bus and they should not have traffic ready to be sent. This will allow the sync generating station to take control of the bus quickly after power failure or initialization.

The failure of a timer or the microprocessor could disable a station's ability to transmit or allow it to transmit in a reserved space. However, there is no provision for fault tolerance in either system. If this feature is necessary, it can be added by the addition of either redundant timer hardware or the addition of a second terminal that will replace the primary terminal if it fails.

4.4 EXPANSIONS AND MODIFICATIONS

Selectable timer values could be used to increase the versatility of the system for a given application. This would allow the system to be reconfigured dynamically to suit known communications traffic pattern changes. An example application would be the change in the communications requirements of a multi-stage launch vehicle after the dropping of a spent stage.

The loading of timer values could be accomplished by several methods. A set of timer values may be stored in ROM and loaded into the timers by the local host at the reception of a particular Ethernet packet. The timer values could also be placed in a broadcast Ethernet packet. The use of a minimum length packet for the sync command would allow for 46 data bytes to be used for command purposes. The use of these bytes to dynamically change various stations timers would allow the system to be very flexible. The sync packet size could be extended to allow more timer values to be set but this is a trade-off with the available bandwidth.

The use of sixteen bit timers requires that two data bytes be sent for each timer count modification. Since the Tc and Td timers must be the same for all stations, only the Ta and Tb timers are unique to each station. Therefore, four bytes are

required for each station. The stations could be assigned four byte slots in the sync packet as shown in Figures 4.11 if the timer values are to be reset at each sync. If only a few stations' timers will be changed, the station addresses for the stations to be modified could be imbedded in the packet as shown in Figure 4.12. Another alternative is to send a mimimum length packet that contains a reference to previously stored timer values as shown in Figure 4.13. Again this is a trade-off between flexibility and bandwidth. The imbedding of addresses would be best if only a small percentage of the total stations are to be changed. The assigned slot method would be best if the access rights of a majority of the stations are to be changed at each sync. This method would be difficult to implement for a large system, however, due to the large number of data words required to change all system timer values.

The use of a microprocessor to control a terminal's timers or to serve as the timer system would allow the terminal to be very versatile with the use of many pattern changes controlled by these processors. The processor could use the 46 bytes of data contained in the minimum Ethernet packet to indicate which set of timer values should be used. These values might have been stored in ROM prior to system power-up or could have been loaded into RAM from previous command packets.

The use of dynamic reassignment changes the MFA protocol from a TDMA (virtual token)/free access system to a command/response/free access system. The sync generating station serves as the bus controller and should be the communications terminal of the central controller. This station might use two transmit and receive addresses. The broadcast address could be used for receiving as in all Ethernet stations and also for transmitting the sync signal. The broadcast transmit address would be used as a flag by the other terminals to indicate that this packet

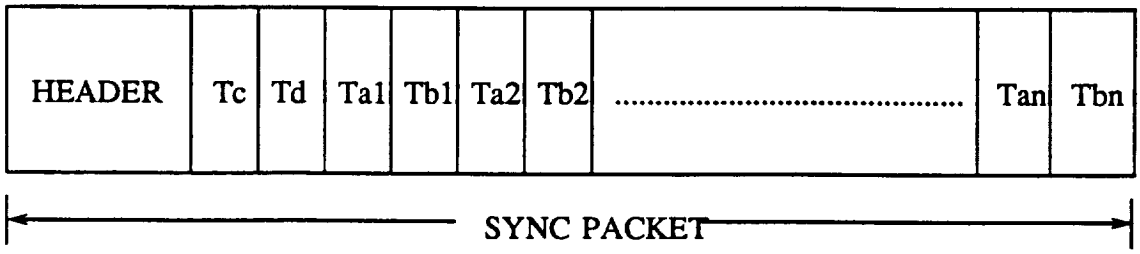


FIGURE 4.11 MFA Sync Packet Implementation for Timing Change on each Sync

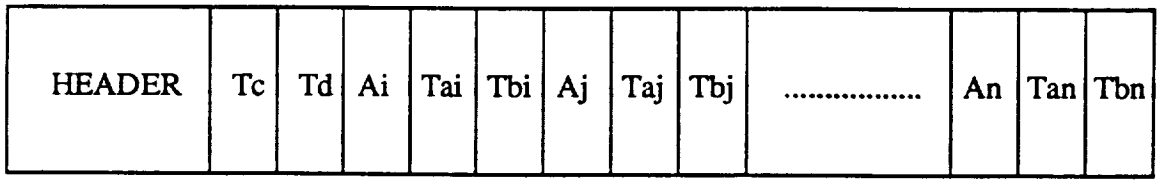


FIGURE 4.12 MFA Sync Packet Implementation for Limited Number of Timing Changes

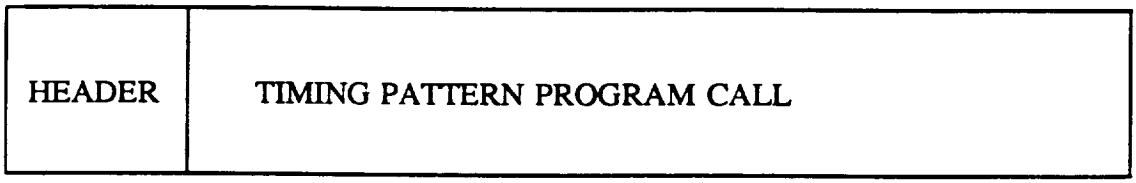


FIGURE 4.13 MFA Sync Packet Implementation for Stored Timing Pattern System

was the sync packet. The unique address would be used for normal communications. The unique address would be used by other stations to address the central controller without using broadcast messages.

Another useful feature that would be easy to implement is to allow the station that just received a packet to transmit an acknowledgment during the transmitters reserved time. This may be required in some systems but has been shown to drastically reduce throughput in long systems.

4.4.1 ADDITIONAL STATION TYPES

Limited: Stations that will only have Poisson distributed traffic and will not require assigned reserved slots may be implemented with only the Tc timer if sync is used with each cycle or the Tc and Td timers if sync is sent only after each group of n cycles. These stations will have no reserved access and will not interfere with the reserved access period. This type of station could be used for a computer system that does not serve as a controller or for other stations that have no time critical access requirements.

Pure: Pure Ethernet stations could be added for short term use as long as their possible contention with reserved stations could be tolerated. These stations should be limited in number and should be used only for initial installation and traffic analysis. These stations can then be replaced with MFA or limited stations after the station's traffic generation pattern is known. The addition of reserved slots for new stations will require the modification of at least the Tc timer in each station. This could be accomplished by network messages if the timers are programmable but will require hardware changes to the timers if they are not programmable.

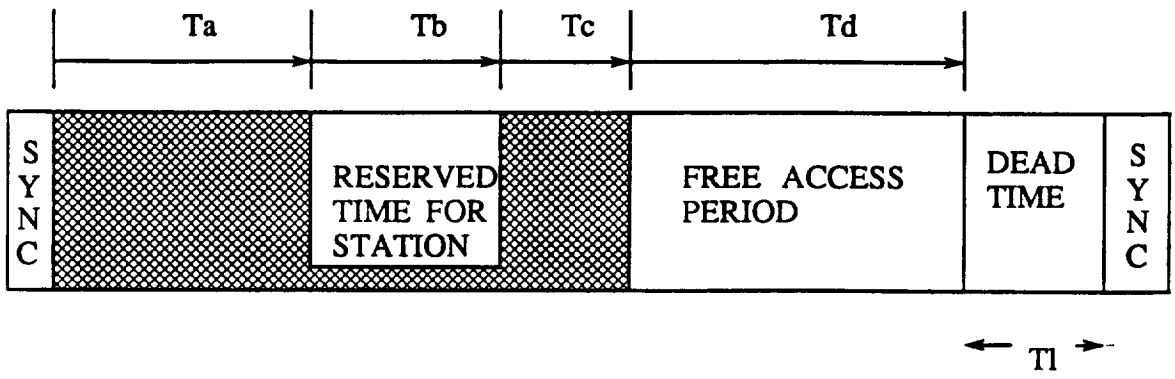
Persistent: A single persistent CSMA station could be added if there was a need for this type station. Possible applications for this station would be a flight controller, system time clock, or the primary MFA station responsible for sync generation.

4.4.2 PROTOCOL MODIFICATIONS

In systems where the Poisson load is known to be high or is known to often interfere with the sync signal, the Td timer could be used to clear the bus so that the sync station would have sole access at the end of each cycle. A timing diagram for this system is shown in Figure 4.14. This implementation throws away bandwidth to assure that the sync signal does not suffer a collision. In most systems a small amount of variance may be allowed in the starting time of each cycle but for heavily loaded or extremely time critical systems this may not be tolerable.

Another possible modification to improve the performance of high priority stations is the use of linear or exponential priority parameters in the backoff algorithm delay calculation. This would improve the chances of a high priority station getting the bus at the expense of the low priority station having longer average delays. The basic scheme in this system is for the high priority stations to use the exponential backoff algorithm time and lower priority stations to use this algorithm with additional time added. This does reduce the average delay of the high priority stations but may dramatically increase the average delay of the low priority stations.

In order to avoid the possibility of reserved time stations being in backoff during the station's reserved access time, only the critical packets should be sent by the reserved time station. A second terminal could be added to handle Poisson traffic from this source. This would not be necessary in a system whose protocol chip



Tl = DEAD TIME REQUIRED TO CLEAR BUS
10 MICROSECONDS MINIMUM

FIGURE 4.14 Td used to Clear Channel for Sync

allowed the exponential backoff timer to be overridden by a clear to send reserved time (CTSRT) signal. It is preferable to place stations with long scheduled packets at the beginning of the reserved time. This should allow stations with small packets to finish their backoff routine before their reserved period. Another reason for placing the long scheduled packets first is that if these stations miss their reserved access times it would be preferable for these stations to be the ones to use the free access period. This will improve performance since one long packet suffers fewer collisions than many small packets as seen from the Ethernet simulation in Chapter 6. The long packet will complete its transmission after it gains control of the channel; while if many small packets must be transferred, they will each have to gain control of the channel.

4.5 COMMAND/RESPONSE MODIFIED FREE ACCESS PROTOCOL

A command/response system could also be implemented by this method. An Ethernet station serving as the bus controller could control all of the clear to send lines through the station's host software or additional hardware and, therefore, would be the bus master. The overhead would be high but the high throughput of the Ethernet system and low

terminal cost could make this system economical for some applications. The command station would probably use a minimum length packet which has 46 bytes of data. The data field would be wasted if only short commands of one or two bytes were sent to only one station. If the commands were broadcast so that all stations could listen and commands for many stations were embedded in the data field, this would significantly reduce the overhead for the system.

4.6 SOFTWARE FOR TIMING ANALYSIS

The use of a timing analysis software package could be very beneficial in assigning reserved access times. Care must be taken that no stations overlap and that there is a minimum of dead time in the reserved access period. There must, however, be some time between individual reserved access periods to allow for propagation delays. A simple program with graphically oriented output would be useful in assigning reserved and free access periods. If all reserved slots were required to be of the same length a reservation table could easily be used to indicate assignments.

4.7 EXAMPLE APPLICATION: LAUNCH VEHICLE AVIONICS COMMUNICATIONS

The communication needs of a launch vehicle with intelligent sensors and distributed intelligence can be described as Poisson traffic with some scheduled traffic. A typical example for a one stage launch vehicle is presented in Table 4.1. The sensors are broken into two classes intelligent and dumb. The intelligent sensors are read periodically by stations needing to know their values but they also act as watchdogs and may generate traffic in situations where critical values are being approached

or for other reasons. The dumb sensors only provide data when requested to by another station in the system.

For this system, using Ethernet, the average delay is the time to transmit two average sized packets[GONS87]. The use of scheduling reduces the average delay for this system from twice the average packet length to near zero. This does not account for the waiting time a Poisson station would experience waiting for the restricted period to pass if it generated a packet during this time. It should be noted in this example that there are 60 periods of 5.76 milliseconds in each second when the channel is reserved.

TABLE 4.1 Data Sources for Launch Vehicle

STATION TYPE	TRANSMISSIONS/SEC	SIZE
1 FLIGHT CONTROLLER	100	150 BYTES
1 NAVIGATION UNIT	60	150 BYTES
1 ENGINE CONTROLLER	60	100 BYTES
100 SENSORS (SCHEDULED)	60/ SENSOR	80 BYTES
40 SENSORS (POISSON)	5/ SENSOR	100 BYTES

TOTAL LOAD = 530×10^3 BYTES / SECOND
= 4.24×10^6 BITS / SECOND

NOTE: OVER NINTY PERCENT OF THE OFFERED LOAD IS SCHEDULED AND THE AVERAGE DELAY OF THE SYSTEM USING STANDARD ETHERNET WITH A FOURTY PERCENT OFFERED LOAD IS TWO AVERAGE PACKET TRANSMISSION TIMES. THIS SYSTEM WITH SCHEDULING WILL HAVE LESS THAN ONE AVERAGE PACKET TRANSMISSION TIME FOR THE AVERAGE PACKET DELAY.

5 MATHEMATICAL ANALYSIS OF PROTOCOL

In this section, a mathematical model is developed to show the improvements that the Modified Free Access (MFA) protocol has for a local area network with scheduled loads. We will use established mathematical models of CSMA and CSMA/CD protocols from Tobagi's work as the basis of a model for MFA.[TOBA87]

5.1 ETHERNET MODEL

The Ethernet model is based on the models described in the paper, "Performance Analysis of Carrier Sense Multiple Access with Collision Detection" by Fouad A. Tobagi and V. Bruce Hunt. [TOBA87] The analysis considers such factors as capacity and the throughput–delay performance of a local area network (LAN) employing Carrier Sense Multiple Access with Collision Detection (CSMA/CD). It can be used to determine the performance of a network as a function of average retransmission delay and collision recovery time. A source for comparison for the results of Tobagi's model is the data given in "Measured Performance of the Ethernet" by Timothy A. Gonsalves. [GONS87] Gonsalves' paper discusses the difficulty in mathematically modeling a LAN using the Ethernet protocol due to the binary exponential backoff algorithm and the physical distribution of the stations in the LAN being modeled. The measurements taken by Gonsalves were of LANs operating at 3 and 10 Mbit/sec. It is shown that the Ethernet protocol achieves high throughput when the packets are long. The correlation between the measurement of the actual LAN and the predictions based on analytical models ranged from good, to poor, depending on the complexity of the model.

5.2 NOTES ON MODELING CSMA/CD PROTOCOLS

(1) In a non-persistent CSMA/CD protocol, a station with a packet ready for transmission senses the communication channel and proceeds as follows:

(i) If the channel is idle, then the station initiates transmission of the packet.

(ii) If the channel is busy, then the station schedules the transmission at some later time, and tries again.

(iii) If a collision is detected during transmission, the transmission is aborted and transmission of the packet is scheduled for some later time.

(2) In a P-persistent protocol, P is the probability that a station will transmit when it senses the channel is open. If $P = 1$, then if a station finds the channel busy it will transmit as soon as the channel becomes free, similar to non-persistent CSMA/CD. If $P \neq 1$, then when a station senses the channel is idle, there is only a P probability that it will transmit and a 1-P probability that it will delay τ seconds (where τ is the end-to-end propagation delay) before taking any further action. After the delay of τ seconds, the station will again either transmit with probability P, or wait another τ seconds with probability 1-P, and so on until it either transmits or senses that the channel is busy in which case the station schedules transmission of the packet at some later time.

If each of the rescheduling delays of the P-persistent and the non-persistent protocols are set to integer multiples of τ seconds, and if the delays were geometrically distributed with parameter P (i.e., $n\tau = P(1-P)^{n-1}$, $n \geq 1$), then P-persistent and non-persistent protocols are identical.

(3) Given that a transmission is initiated on an EMPTY channel, it will take at most one end–to–end propagation delay, τ, for the first packet to reach all stations on the channel. After that, the channel is sensed busy for as long as data is transmitted.

(4) Some station will possibly sense the channel is idle when in fact another station has begun transmission. Due to propagation delay the signal will not reach the previously idle station before it begins transmission. When this happens, the station will begin transmitting and its signal will interfere (collide) with the signal already on the channel. In this case, it will take (at most) τ seconds before interference reaches all devices.

(5) ξ is the time that it takes a station to detect interference once the collision front reaches the station's connection to the bus. ξ depends on the implementation of the system. ξ is approximately one bit transmission time for most implementations.

(6) For the first packet to reach all stations, at most τ seconds is needed. For a station to detect interference, at most $\tau+\xi$ seconds are needed. And, if ζ seconds are needed for collision consensus, then at most α seconds are needed until the channel is sensed idle by all stations. This means that, given a collision occurs, then the time required for all devices to cease transmission, γ, is $2\tau+\xi+\zeta$, and at most it will take $\alpha=3\tau+\xi+\zeta$ seconds until the channel is again sensed idle by all stations.

(7) Simplifying Assumptions:

(a) The time axis is divided into "slots" τ seconds long.

(b) Transmission may only begin at the beginning of a slot.

(c) All stations on the channel are time synchronized.

(d) When any station becomes ready in some slot, it senses the channel during that slot and then operates according to the CSMA/CD protocol.

5.3 CSMA/CD CHANNEL CAPACITY

The model is considered as an infinite population model. The sequence of transmission periods are the considered sequences of events and such events include successful transmission, unsuccessful transmission, and idle periods. All stations are also considered collectively from an independent Poisson source and τ is arbitrarily large. The following are some properties of the model:

(a) A transmission period followed by an idle period is called a "cycle."

(b) Since an infinite population model is assumed, then all cycles are statistically identical.

(c) Since a Poisson source is assumed, the probability that the number of transmissions, X is equal to k is given as: $P\{X=k\} = \frac{e^{-g}g^k}{k!}$ where g is the rate at which stations become ready during a slot in stations/slot-time.

(d) The length of a packet, i.e., the number of slots required for its transmission, is denoted by T. A successful transmission period is, therefore, of length T + 1 slots. In the case of a collision, the length of a transmission period is $\gamma + 1$.

(e) The probability of zero new transmissions is e^{-g}.

(f) The probability of one new transmission is ge^{-g}.

(g) The probability of at least one new transmission is one minus the probability of zero new transmissions, or $1-e^{-g}$.

(h) The conditional probability of exactly one new transmission given the occurrence of one or more new transmissions is $P_s = \frac{ge^{-g}}{1-e^{-g}}$.

(i) The average transmission period is:

$$\begin{aligned}
\overline{TP} &= (T+1)P_s + (\gamma+1)(1-P_s) \\
&= TP_s + P_s + \gamma(1-P_s) + (1-P_s) \\
&= TP_s + \gamma(1-P_s) + 1 \\
&= T\left(\frac{ge^{-g}}{1-e^{-g}}\right) + \gamma\left(1-\frac{ge^{-g}}{1-e^{-g}}\right) + 1 \\
&= \frac{Tge^{-g}}{1-e^{-g}} + \gamma - \frac{\gamma ge^{-g}}{1-e^{-g}} + 1 \\
&= \frac{Tge^{-g} - \gamma ge^{-g} + \gamma(1-e^{-g}) + 1 - e^{-g}}{1-e^{-g}}
\end{aligned}$$

(j) The idle period may be described by a geometric distribution. The geometric distribution is described by a single parameter, the probability of success, p. For this case, success is when one or more stations generates new traffic and failure is when no stations have new traffic. P is given by the probability of a new transmission which is one minus the probability of zero new transmissions, or $1-e^{-g}$. The geometric distribution with probability of success p has a mean value of 1/p. Therefore, the average idle period is defined by the equation

$$\bar{I} = \frac{1}{1-e^{-g}} .$$

5.4 1-PERSISTENT CSMA/CD THROUGHPUT EXPRESSION:

The following quantities can be calculated given that the probability of generation g is known:

$\overline{B}$ = Average duration of a busy period

$\overline{I}$ = Average duration of an idle period

$\overline{U}$ = Average time during a cycle that the channel is carrying successful transmissions (such periods will be called "successful" periods).

The throughput, S, can be expressed as a function of $\overline{B}$, $\overline{I}$, and $\overline{U}$, specifically $S = \frac{\overline{U}}{\overline{B} + \overline{I}}$. If we let $\gamma = 2\tau$, (that is, $\zeta+\xi=0$) then $\frac{\gamma}{\tau} = 2$, and it follows that:

$$S = \frac{Tge^{-g}}{Tge^{-g} + 2 - 2e^{-g} - 2ge^{-g} + 2 - e^{-g}}$$

$$= \frac{T^{-1}g^{-1}e^{g}Tge^{-g}}{T^{-1}g^{-1}e^{g}Tge^{-g} + 2T^{-1}g^{-1}e^{g} - 2T^{-1}g^{-1}e^{g}e^{-g} - 2T^{-1}g^{-1}e^{g}ge^{-g} + 2tg - T^{-1}g^{-1}e^{g}e^{-g}}$$

$$= \frac{1}{1 + 2T^{-1}g^{-1}e^{g} - 2T^{-1}g^{-1} - 2T^{-1} + 2T^{-1}g^{-1}e^{g} - T^{-1}g^{-1}}$$

$$= \frac{1}{1 + 4T^{-1}g^{-1}e^{g} - 3T^{-1}g^{-1} - 2T^{-1}}$$

Now, if we define $a \equiv \frac{1}{T} = \frac{propagation\ delay}{packet\ transmission\ time}$, then:

$$S = \frac{1}{1 + \left(\frac{4e^{g} - 3}{g} - 2\right)a}$$

$$= \frac{1}{1 + H(g)a}$$

where $H(g) = \frac{4e^g - 3}{g} - 2$. (Note: if $T < 2$, then the system will need $\gamma < 2$ for all devices to stop transmitting. If $T < 2$, then an undetected collision may occur since the transmitting station may complete its transmission before detecting the collision. This situation arises when a station at one extreme of the bus begins transmitting and just before this transmission reaches the other end of the bus, a station at that end begins transmitting, thus yielding a collision. In the time it takes the collision-garbled data to reach the first station at the extreme of the bus, that station may have completed its transmission and therefore not recognize the collision with its packet. This packet will be lost unless a higher level protocol is in use to verify the reception of the packet.

Now, if we let $H = \min_g\{H(g)\}$, then $C(\infty, a) = \frac{1}{1 + Ha}$ when $a \leq 0.5$ and $T \geq 2$. However, if $T < 2$, and $a > 0.5$ then the case will be:

$$P_s = \frac{ge^{-g}}{1 - e^{-g}} \qquad \textit{(as before)}$$

$$\bar{I} = \frac{1}{1 - e^{-g}} \qquad \textit{(as before)}$$

$$\overline{TP} = P_s T + (1 - P_s)\gamma + 1 = 3$$

since T will never be less than γ, and $\gamma = 2$, and therefore $T_{min} = 2$. From this we know that:

$$
\begin{aligned}
S &= \frac{P_s T}{\overline{TP} + \overline{I}} \\
&= \frac{\frac{Tge^{-g}}{1-e^{-g}}}{3 + \frac{1}{1-e^{-g}}} \\
&= \frac{Tge^{-g}}{3(1-e^{-g}) + 1} \\
&= \frac{Tge^{-g}}{2ge^{-g} + 2(1-e^{-g}-ge^{-g}) + 2 - e^{-g}} \\
&= \frac{1}{2ge^{-g}T^{-1}g^{-1}e^{g} + 2(1-e^{-g}-ge^{-g})T^{-1}g^{-1}e^{g} + 2T^{-1}g^{-1}e^{g} - e^{-g}T^{-1}g^{-1}e^{g}} \\
&= \frac{1}{2T^{-1} + 2(T^{-1}g^{-1}e^{g} - T^{-1}g^{-1} - T^{-1}) + 2T^{-1}g^{-1}e^{g} - T^{-1}g^{-1}} \\
&= \frac{1}{2a + \frac{4ae^{g}}{g} - \frac{3a}{g} - 2a} \\
&= \frac{1}{a(2 + H(g))} \\
&= \frac{1}{a(2 + H)}
\end{aligned}
$$

where $H = \min_g\{H(g)\}$, as before.

If we define $q_i(x)$ as the probability that there are i arrivals in x slots, then

$$q_i(x) \approx e^{-gx}\frac{(gx)^i}{i!} \quad .$$

Thus, the probability of exactly one arrival in x number of slots is

$$q_1(x) \approx e^{-gx}\frac{(gx)^1}{1!} = gxe^{-gx}$$

and the probability of exactly zero arrivals in x number of slots is

$$q_0(x) \approx e^{-gx}\frac{(gx)^0}{0!} = e^{-gx} \quad .$$

Given that a transmission of x slots in length is occurring, we know that the probability of one or more new arrivals during the x slots is $1-q_0(x) = 1-e^{-gx}$. Thus the probability of a successful transmission after a transmission of x slots is simply the probability of exactly one arrival given that there were one or more arrivals during the x slots, or

$$P_s(x) = \frac{q_1(x)}{1-q_0(x)} = \frac{gxe^{-gx}}{1-e^{-gx}}$$

which agrees with the result derived earlier. The probability of unsuccessful transmissions is the probability of two or more arrivals, or

$$P_u(x) = 1 - P_s(x) = 1 - \frac{q_1(x)}{1-q_0(x)} = 1 - \frac{gxe^{-gx}}{1-e^{-gx}}$$

The probable length of the busy period, B(x), can be found by

$$B(x) = P_s * term_1 + P_u * term_2$$

where $term_1$ is the probable length of the busy period in the case of a successful transmission, and $term_2$ is the probable length of the busy period in the case of an unsuccessful transmission.

First, we will attempt to compute $term_1$, the probable length of the busy period in the case of a successful transmission. In order to have a successful transmission, we must have $x = T+1$, and, thus, in the case of successful transmission:

$T+1$ = length of a successful transmission busy period,

$1-q_0(T+1)$ = probability of one or more arrivals in the $T+1$ period,

$B(T+1)$ = average of the remainder of the busy period in which a successful transmission occurs.

Since the number of arrivals in the $T+1$ period will be the weight of the average of the remainder of the busy period, then

$$term_1 = T + 1 + [1 - q_0(T+1)]B(T+1)$$

is the probable length of time of the busy period in the case of a successful transmission. It is important to note that this is a recursive equation in that after the $T+1$ slot successful transmission is completed the average remainder of a $T+1$ transmission must be added if a new arrival occurs during the transmission. The $B(T+1)$ term then expresses the remainder after this transmission and contains the appropriate probabilities to weight the average time for successful and unsuccessful transmissions.

In order to compute $term_2$, we must consider the case of an unsuccessful transmission. For unsuccessful transmission we let:

$\gamma + 1$ = length of a busy period in which a collision occurs,

$1 - q_0(\gamma+1)$ = probability of one or more arrivals in the $\gamma+1$ period,

$B(\gamma+1)$ = average of the remainder of the busy period in which an unsuccessful transmission occurs.

By the same reasoning used to compute $term_1$, we have

$$term_2 = \gamma + 1 + [1 - q_0(\gamma+1)]B(\gamma+1)$$

which is the average length of the busy period in the case of an unsuccessful transmission.

Substituting the definitions of P_u, P_s, $term_1$ and $term_2$ into the equation for the average length of the remainder of the busy period, B(x) yields:

$$B(x) = \frac{q_1(x)}{1 - q_0(x)}\{T + 1 + [1 - q_0(T+1)][B(T+1)]\}$$

$$+ \left[1 - \frac{q_1(x)}{1 - q_0(x)}\right]\{\gamma + 1 + [1 - q_0(\gamma+1)][B(\gamma+1)]\}$$

If we consider only a successful transmission, then the remainder of the busy period

becomes

$$B(T+1) = \frac{q_1(T+1)}{1-q_0(T+1)}\{T+1+[1-q_0(T+1)][B(T+1)]\}$$

$$+\left[1-\frac{q_1(T+1)}{1-q_0(T+1)}\right]\{\gamma+1+[1-q_0(\gamma+1)][B(\gamma+1)]\}$$

and similarly, in the case of an unsuccessful transmission, the remainder of the busy period becomes

$$B(\gamma+1) = \frac{q_1(\gamma+1)}{1-q_0(\gamma+1)}\{T+1+[1-q_0(T+1)][B(T+1)]\}$$

$$+\left[1-\frac{q_1(\gamma+1)}{1-q_0(\gamma+1)}\right]\{\gamma+1+[1-q_0(\gamma+1)][B(\gamma+1)]\}$$

Since only one slot time is required to initiate a transmission, (whether successful or not), $\overline{B} = B(1)$.

Now, we can see that

$$B(T+1)[1-q_1(T+1)] = \frac{q_1(T+1)}{1-q_0(T+1)}(T+1)$$

$$+\left(1-\frac{q_1(T+1)}{1-q_0(T+1)}\right)\{\gamma+1+[1-q_0(\gamma+1)]B(\gamma+1)\}$$

$$B(\gamma+1)[1-q_1(\gamma+1)] = \frac{q_1(\gamma+1)}{1-q_0(\gamma+1)}\{T+1+[1-q_0(T+1)]B(T+1)\}$$

$$+\left(\frac{1-q_0(\gamma+1)-q_1(\gamma+1)}{1-q_0(\gamma+1)}\right)\{\gamma+1+[1-q_0(\gamma+1)]B(\gamma+1)\}$$

$$B(\gamma+1)[1-q_1(\gamma+1)-1+q_0(\gamma+1)+q_1(\gamma+1)] = \frac{q_1(\gamma+1)}{1-q_0(\gamma+1)}\{T+1+[1-q_0(T+1)]B(T+1)\}$$

$$+\left(\frac{1-q_0(\gamma+1)-q_1(\gamma+1)}{1-q_0(\gamma+1)}\right)(\gamma+1)$$

and

$$B(T+1)[1-q_1(T+1)] = \frac{q_1(\gamma+1)}{1-q_0(\gamma+1)}(T+1) + \left(1-\frac{q_1(T+1)}{1-q_0(T+1)}\right)(\gamma+1)$$

$$+\left(1-\frac{q_1(\gamma+1)}{1-q_0(\gamma+1)}\right)[1-q_0(\gamma+1)]\left(\frac{1}{q_0(\gamma+1)}\right)$$

$$\left[\left(\frac{q_1(\gamma+1)}{1-q_0(\gamma+1)}\right)\{T+1+[1-q_0(T+1)]B(T+1)]\} + \left(\frac{1-q_0(\gamma+1)-q_1(\gamma+1)}{1-q_0(\gamma+1)}\right)(\gamma+1)\right]$$

We now have B(T+1) in terms of known parameters and may solve for other B(X) values. But, the average duration of a busy period, $\overline{B}$, is B(1). That means that

$$B(1) = \frac{q_1(1)}{1-q_0(1)}\{T+1+[1-q_0(T+1)][B(T+1)]\}$$

$$+\left[1-\frac{q_1(x)}{1-q_0(x)}\right]\{\gamma+1+[1-q_0(\gamma+1)][B(\gamma+1)]\}$$

where again, B(T+1) and B(γ+1) are found from the equations:

$$B(T+1) = \frac{q_1(T+1)}{1-q_0(T+1)}\{T+1+[1-q_0(T+1)][B(T+1)]\}$$

$$+\left[1-\frac{q_1(T+1)}{1-q_0(T+1)}\right]\{\gamma+1+[1-q_0(\gamma+1)][B(\gamma+1)]\}$$

and

$$B(\gamma+1) = \frac{q_1(\gamma+1)}{1-q_0(\gamma+1)}\{T+1+[1-q_0(T+1)][B(T+1)]\}$$

$$+\left[1-\frac{q_1(\gamma+1)}{1-q_0(\gamma+1)}\right]\{\gamma+1+[1-q_0(\gamma+1)][B(\gamma+1)]\}$$

The utilization U(x) may be found in a similar manner. This is the average remaining time the channel will be passing packets given that a transmission of length x was just completed. During a successful transmission the value of utilization is T. The equation for U(x) is given by the recursive equation:

$$U(x) = \frac{q_1(x)}{1 - q_0(x)}\{T + [1 - q_0(T + 1)][U(T + 1)]\}$$

$$+ \left[1 - \frac{q_1(x)}{1 - q_0(x)}\right]\{[1 - q_0(\gamma + 1)][U(\gamma + 1)]\}$$

5.5 CSMA/CD CHANNEL DELAY ANALYSIS

A non-persistent protocol is considered in the following discussion. The properties of the model are as follows:

(a) The model consists of M stations (that is, a finite population model).

(b) At any given time, a station is either in the BACKLOGGED state or in the THINKING state. If the station is in the THINKING state, it generates a new packet in a slot with probability σ and transmits the packet provided that the channel is sensed idle. On the other hand, if the station is in the BACKLOGGED state, its packet either experienced a collision during an attempt at transmission or its transmission was blocked due to the channel being busy.

(c) A BACKLOGGED device remains in that state until it completes a successful transmission of its packet at which time it switches back into the THINKING state.

(d) The rescheduling delay of a BACKLOGGED packet is geometrically distributed with a mean of $\frac{1}{\nu}$ slots. This can be modeled as each BACKLOGGED station sensing the channel in the current slot with a probability ν. If the channel is sensed clear, then the station can leave backlog, if not the station remains in backlog. It is important to note that this policy is different than the Ethernet backlog policy where a station must wait a designated number of slots and then may sense the channel.

The transition probability matrix between consecutive points, P, is generated by considering the system to be an embedded Markov chain. The P matrix is the state transition matrix between the embedded points Π_i which are the probabilities that the system has i stations in backlog. The matrix P can be expressed as the product of several single slot transition matrices. Tobagi in his paper indicates that $P = S(Q)^{T+1}J + F(Q)^{\gamma+1}$, where the matrices S, F, J, and Q are defined as follows.

The S matrix indicates the probability that after a successful transmission there will be k stations in backlog if the system started with i stations in backlog at the beginning of the transmission. The matrix S_{ik} is defined as follows:

$$S_{ik} = \begin{cases} 0, & k < i \\ \dfrac{(1-\sigma)^{M-i}[i\nu(1-\nu)^{i-1}]}{1-(1-\nu)^i(1-\sigma)^{M-i}}, & k = i \\ \dfrac{(M-i)\sigma(1-\sigma)^{M-i-1}[1-\nu]^i}{1-(1-\nu)^i(1-\sigma)^{M-i}}, & k = i+1 \\ 0, & k > i+1 \end{cases}$$

In Tobagi's second equation for S_{ik}, the transmission comes from a station in backlog and given there were i stations in backlog at the beginning of the transmission there will be i–1 stations in backlog at the completion. Therefore, this equation is for

$k = i-1$ not $k = i$ as indicated. The third equation is for a successful transmission from a thinking station given that there were i stations in backlog when the packet was generated by a station in the thinking state. At the completion of this transmission there will be no additions to the backlog because this would require another station to have generated traffic resulting in a collision and an unsuccessful transmission, therefore, $k = i$ not $k = i+1$.

The probability that the successful transmission comes from backlog is given by the probability that no thinking station generates new traffic multiplied by the probability that only one backlogged station senses the channel conditional on there being a transmission. The probability that none of the M–i thinking stations generates new traffic is $(1-\sigma)^{M-i}$. The probability that only one of the i backlogged stations leaves backlog is $i\nu(1-\nu)^{i-1}$. This is the probability of one station leaving backlog multiplied by the probability that i–1 stations do not leave backlog multiplied by i which is the number of ways this event can happen. The probability is conditional on there being a transmission. This probability is (1 – probability of no transmissions). The probability of no transmissions is given by the probability that no backlogged station senses the channel and no thinking station generates new traffic. This is given by the expression $(1-\sigma)^{M-i}(1-\nu)^{i}$. Since the transmission is successful the backlog must either remain constant or decrease by one. There can only be one transmission and it can come from a backlogged station or a thinking station.

The F matrix indicates the probability that there will be k stations in backlog after an unsuccessful transmission attempt provided that there were i stations at the beginning of the transmission. The matrix F_{ik} is described as follows:

$$F_{ik} = \begin{cases} 0, & k<i \\ \dfrac{(1-\sigma)^{M-i}[1-(1-\nu)^i - i\nu(1-\nu)^{i-1}]}{1-(1-\nu)^i(1-\sigma)^{M-i}}, & k=i \\ \dfrac{(M-i)\sigma(1-\sigma)^{M-i-1}[1-(1-\nu)^i]}{1-(1-\nu)^i(1-\sigma)^{M-i}}, & k=i+1 \\ \dfrac{\binom{M-i}{k-i}(1-\sigma)^{M-k}\sigma^{k-i}}{1-(1-\nu)^i(1-\sigma)^{M-i}}, & k>i+1 \end{cases}$$

The probability that k < i is zero since no station can relieve itself of a backlogged packet when a collision occurs. This, however, does not cover the contingency that a packet is discarded after a given number of attempts. The probability that the number of stations backlogged remains constant requires that no thinking station generate new traffic and that at least two stations leave backlog and attempt to transmit. The probability that none of the (M–i) thinking stations generate new traffic is as given above. The probability that two or more stations leave backlog is the probability that at least one station leaves backlog given by $[1-(1-\nu)^i]$ minus the probability that only one station leaves backlog as given above. This is again conditional on there being a transmission. The probability that k = i + 1 after an unsuccessful transmission is given by the probability that exactly one thinking station generates new traffic and at least one backlogged station attempts to leave backlog. This is given by $[(M-i)\sigma(1-\sigma)^{M-i-1}][(1-(1-\nu)^i]/[1-(1-\sigma)^{M-i}(1-\nu)^i]$. Again this is conditional on there being a transmission. The probability that k > i + 1 is given by the probability that more than one station generates new traffic. In order to use this expression in determining the total system we wish to know the value of k after the unsuccessful attempt. It is, therefore, necessary to determine this probability for each value of k. This is dependent

only on the number of new stations generating traffic since two or more of these will result in a collision regardless of the activity of the backlogged stations. The probability that k stations will leave backlog is given by:

$$\frac{\binom{M-i}{k-i}(1-\sigma)^{M-k}\sigma^{k-i}}{1-(1-\nu)^{i}(1-\sigma)^{M-i}}, \qquad k>i+1.$$

This is the probability that M–k stations do not leave backlog multiplied by the probability that k–i stations generate new traffic conditional on there being a transmission.

The Q matrix is used to indicate the probability of thinking stations generating new traffic during a collision or a successful transmission and therefore going into backlog. The J matrix indicates that the backlog is decreased on a successful transmission. Tobagi's Q and J matrices are, therefore, described as follows:

$$Q_{ik} = \begin{cases} 0, & k<i \\ \binom{M-i}{k-i}(1-\sigma)^{M-k}\sigma^{k-i}, & k \geq i \end{cases}$$

$$J_{ik} = \begin{cases} 1, & k = i-1 \\ 0, & \textit{otherwise} \end{cases}$$

However, the S matrix indicates the probability of a transmission from backlogged and thinking stations. Both of these must be included in the state transmission matrix P_{ij}. This results in the equation defining P_{ij} to be $P = S(Q)^{T+1} + F(Q)^{\gamma+1}$, where the matrices F and Q are defined as above and the S matrix with indices corrections is described as follows:

$$S_{ik} = \begin{cases} 0, & k < i-1 \\ \dfrac{(1-\sigma)^{M-i}[i\nu(1-\nu)^{i-1}]}{1-(1-\nu)^{i}(1-\sigma)^{M-i}}, & k = i-1 \\ \dfrac{(M-i)\sigma(1-\sigma)^{M-i-1}[1-\nu]^{i}}{1-(1-\nu)^{i}(1-\sigma)^{M-i}}, & k = i \\ 0, & k > i \end{cases}$$

The average channel throughput, $\overline{S}$, is given by the average busy time divided by the average cycle time. This may be expressed as follows:

$$\overline{S} = \frac{\sum_{i=0}^{M} \Pi(i) Ps(i) T}{\sum_{i=0}^{M} \Pi(i)\left\{\frac{1}{1-\delta i} + 1 + Ps(i)T + [1 - Ps(i)]\gamma\right\}}$$

where Ps(i), the probability of transmission being successful given that there is a transmission, is given by:

$$Ps(i) = \frac{(M-i)\sigma(1-\sigma)^{M-i-1}[1-\nu]^{i} + (1-\sigma)^{M-i}[i\nu(1-\nu)^{i-1}]}{1-(1-\nu)^{i}(1-\sigma)^{M-i}}$$

The average number of stations in backlog is given by the sum of stations in backlog during each slot of a cycle divided by the average length of a cycle. The sum of backlogs may be divided into two parts when the channel is idle or when the channel is active. The average value of the backlog is given by the expression:

$$\overline{N} = \frac{\sum_{i=0}^{M} \Pi(i)\left[\frac{i}{1-\delta i} + A(i)\right]}{\sum_{i=0}^{M} \Pi(i)\left\{\frac{1}{1-\delta i} + 1 + Ps(i)T + [1 - Ps(i)]\gamma\right\}}$$

where $\delta_i = (1-\gamma)^i(1-\sigma)^{M-i}$ and A(i) is the expected sum of backlogs over all slots in the busy period after an idle period that has i stations in backlog. This is given by the expression:

$$A(i) = \sum_{j=i}^{M} j\left[S \sum_{l=0}^{T} Q^l + F \sum_{l=0}^{\gamma} Q^l \right]_{ij}$$

The equation for A(i) should include the station that is transmitting. This may be done by considering a thinking station that begins transmitting to be in backlog until it completes transmission. This would follow Tobagi's original expressions for Sik. By Little's result the average packet delay, $\overline{D}$, may be expressed by: $\overline{D} = \overline{N}/\overline{S}$ which may be expressed by:

$$\overline{D} = \frac{\sum_{i=0}^{M} \Pi(i)[\frac{i}{1-\delta i} + A(i)]}{\sum_{i=0}^{M} \Pi(i) Ps(i) T}$$

5.6 PROGRAMMING AND RESULTS

A FORTRAN computer program was written to generate the elements of the P matrix for any given set of parameters according to the method suggested in the paper by Tobagi and Hunt. [TOBA87] The matrices involved contain many small numbers and the Q matrix when raised to the T + 1 power quickly approaches all zeroes. This led to many of the intermediate products being zero yielding NaN (IEEE floating point representation of Not a Number) responses to some calculations. The system usually provided good results if ν was less than 0.15. For Ethernet a station whose packet has collided will sense the channel with probability $\nu = 0.5$.

Of major concern was the output of the program run with the following param-

eters: M (number of stations) = 20, T (packet transmission time in slots) = 16, σ (probability of generation for each station) = 0.1/(20*16) = 3.12500e-04, γ = 2, ν = 0.1. $X_{pp}(i,k)$ is the probability that given a transmission begins with i stations in backlog there will be k stations in backlog at the completion of the transmission. Pi(i) is the likelihood that their will be i stations in backlog. Throughput is normalized to one were 1 would be 10 megabits/second. The value of 0.09987 is very close to the offered load of 0.1. Delay is normalized to T, the packet length, which is 16 slots for the data presented below.

xpp(0,0) = 0.89654910564423
xpp(0,1) = 9.5526792109013d-02
xpp(1,0) = 0.84863811731339
xpp(1,1) = 0.13602685928345
xpp(1,2) = 1.4704237692058d-02
xpp(2,1) = 0.83170044422150
xpp(2,2) = 0.15561614185572
xpp(2,3) = 1.2293432839215d-02
xpp(12,11) = 0.49854353070259
xpp(12,12) = 0.49486445449293
xpp(20,19) = 0.30593270063400
xpp(20,20) = 0.69406725803856

pi(0) = 0.88266480634446
pi(1) = 0.10759882424534
pi(2) = 1.0393591330471d-02
pi(3) = 4.4644482246755d-04
pi(4) = -2.1841781717240d-05
pi(18) = -8.9821003052080d-05
pi(19) = -9.5473198021788d-05
pi(20) = -1.0151982147586d-04

throughput= 9.98713e-02
delay= 1.2809 packet transmission times

The sums of the elements of the columns of P_{ij} all equal one and this indicates that the transition matrix calculations are valid. Some of the components of the S_{ik},

F_{ik} and Q_{ik} matrices and values of $P_s(n)$ were checked against values calculated on a HP41-CV. The values were extremely close indicating that the Fortran routines were correct. The P_{ij} matrix for this as well as several other runs are given in Appendix C.

There are several trends that should be noticed. First, it is interesting to note that for i less than 12, that system is likely to reduce the number in backlog while for i = 12 the system is almost as likely to keep 12 in backlog as to reduce to 11 and at i = 20 the system is much more likely to keep 20 in backlog than to reduce to 19 in backlog. This is due to the value of ν and looking at the other data it seems that the probability that a backlogged station will sense the channel (ν) must be very small, i.e. less than 0.15 for stability and proper operation. The value 0.01 indicates that on average it will take 100 slots for a station to leave backlog. This value is very high for Ethernet where eight collisions are required for the backoff period to be normally distributed over the range zero to 255 slots. The problem in modeling this system is that when a large number of stations become backlogged, the exponential backoff algorithm allows colliding stations to have longer and longer backoff periods until the system clears. This is very hard to express in a mathematical model of the system where only the present state of the machine is kept because the previous number of collisions is required to be known for the backoff algorithm to work properly. The attempted use of an average ν did not yield good results, however, if a $\nu(M,i,\sigma)$ could be found, it would improve the accuracy of the mathematical representation.

Another point of interest is that it is most likely that there will be no stations in backlog and the probability of having i stations in backlog decreases as i increases. The probability goes negative for pi(4) and grows in magnitude for increasing i. This is attributed to accumulating error from the matrix multiplications and additions where the matrices contain many small numbers. An idea around this is to estimate

the Π_i for large i by an estimate of a previous Π_i. The justification for this is that the stationary probability that the system will have i stations in backlog should be less than the stationary probability that the system will have k stations in backlog provided that i > k. This holds true for small loads, however, as the offered load increases the probability that stations will be backlogged increases. At an offered load at or above 100 percent all M stations could be backlogged. A major concern for the system is that p(20,20) is greater than 0.5 and usually between 0.7 and 1.0. This indicates the system is likely to remain with these stations in backlog.

Another attempt was made to model the system mathematically using the technique presented in "Modeling and Analysis of Computer Communications Networks" by Jeremiah F. Hayes as a guide. [HAYE84] The attempt to generate a delay versus offered load plot for CSMA/CD using Equation (8.30), on page 230 resulted in large errors compared to Gonslave's measured data and the models values of delay.

$$\overline{D} = \overline{m} + T\left(\frac{1}{2} + \frac{1}{\nu}\right) - \left\{\frac{1 - e^{-\lambda\tau}}{2[B(0)\nu - (1 - e^{-\lambda T})]}\right\}\left[\frac{2}{\lambda} - 2T\right] + \frac{\lambda[\overline{m^2} + 2\overline{m}\tau + + \tau^2 + 2(\overline{m} + T)(\frac{T}{\nu}) + \frac{T^2(2-\nu)}{\nu^2}]}{1 - \lambda(\overline{m} + \tau - \frac{T}{\nu})}$$

The resulting plot is shown as Figure 5.1. The general shape of the resulting curve was similar to the other performance curves but, unfortunately, did not match the data produced through the simulation procedure. This may be due to the fact that Equation (8.30) is a simplified expression and does not utilize as many parameters as the simulation.

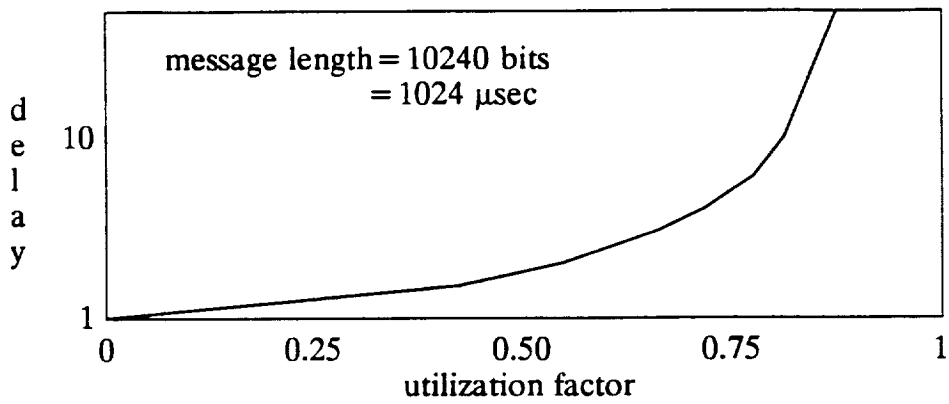


Figure 5.1 Output of Delay Equation

5.7 APPLICATION OF MATH MODEL TO MFA PROTOCOL

The math models that have been presented are for CSMA/CD networks with a fixed channel sensing probability for stations in backoff. Though the backoff algorithm is not the same as that of the Ethernet or MFA protocols the first model is a good estimate of their performance. For low offered loads, when the average number of stations in backoff is low, the model should give a rough approximation of performance.

The model can be used to compare performance between the Ethernet and MFA protocols. The MFA protocol reduces σ the probability of generating new traffic while only slightly increasing average packet length. The scheduled period is considered to be one long packet and there will be no collisions during its transmission. For a long network the propagation delay times must be included in the scheduled period. The MFA protocol will still out perform Ethernet because during the scheduled period the channel will run at maximum performance with no collisions. The MFA protocol is for a system with scheduled data and its performance will be drastically reduced if stations do not utilize their reserved periods. However, the scheduling of a slot for particular transmissions may be required for some applications.

6 PROTOCOL SIMULATORS AND OUTPUT

6.1 PROTOCOL SIMULATORS

A computer simulation program was written to verify the characteristics of the protocol. Since the MFA protocol is similar to the Ethernet protocol, an Ethernet simulator was written and verified first. The simulation parameters are for a short network as would be found in an aircraft or other systems. However, the standard Ethernet packet length's, preamble length and other parameters were kept. The channel clearance time after a transmission was assumed negligible. The system is slotted and only allows transmissions to begin at the slot edge. Collisions take one slot and idle time is accounted for in the packet transmission time. The simulator is for use as a comparison tool but with proper use of the variables it can be used to model systems with varying distributions of offered load and unbalanced traffic offerings from the terminals.

6.2 SIMULATOR OUTPUT DESCRIPTION

The following are sample plots of the data taken from the simulators. The nomenclature used to identify the simulation the data is taken from is as follows. E'A'o indicates that the data is the output of the Ethernet simulator with a Poisson load. There are four of these namely eao, ebo, eco, and edo. The a, b, c and d designations indicate packet size with the values as follows:

a = packet size of 1 slot time (minimum allowable packet size)

b = packet size of 8 slot times (average packet size)

c = packet size of 64 slot times (maximum allowable packet size)

d = packet size of 3 slot times (packet used for comparison to Gonsalve's results).

The outputs for simulators with scheduled loads are of the form o##A#. The letter o was used as an identifier that this is the output of a simulation as opposed to the Pascal or executable code for the given simulation. The first number indicates the type of simulation with numbers 1 through 3 indicating the following:

1 = Scheduled load using Ethernet protocol

2 = Scheduled load using MFA protocol with the Sync Packet contending for the channel

3 = Scheduled load using MFA protocol with the Sync Packet non–contending.

The second number indicates the percent scheduled data. This is taken as percent of 10 megabit capacity that is scheduled. The simulations are run for offered loads of 10 through 100% by tens with offered loads less than the amount scheduled considered to have ideal characteristics. The scheduled loads used in the trials were 10, 20, 30, 50 and 70%. The letter is used as it was for the Ethernet simulator to indicate packet size. For the simulations only the packet sizes of a, b and c were used. The last number indicates the length of the scheduled cycle. The numbers used were 2, 3 and 4 indicating 100, 1000 and 10,000 slot times respectively.

6.3 SIMULATOR OUTPUT PLOTS

The plots follow the expected trends with better performance as the scheduled load component is increased for the MFA protocols and worse performance for the Ethernet protocol as scheduled load is increased. The verification of the protocol against Ethernet throughput is excellent with the delay and other parameters being good.

The plots for Ethernet follow the expected patterns except for the mean delay versus throughput curve for eco. The deviation is only for offered loads of 0.2 thru 0.6 and occurred on both machines. The random number generators for the machines

yield different sequences so it seems unlikely that this is due to the traffic pattern generated by the random number generator. The simulations were run to require at least 5,000 transmissions to be completed and usually 50,000 transmissions were required. This should result in an average value for the parameters. The initial and final values of the eco plot follow the expected path so it appears that the either the delay for a given throughput does increase rapidly or there is some possibility that the random number generators have faults for these very small numbers. Histograms were made of the random number outputs for the total range of zero to one as well as plots for smaller ranges near zero. The appropriate number of points were found to lie in each range. There were a few minor inconsistencies on the order of .1 percent but this should not have been a problem.

The output from the simulator is presented in several curves. The performance as a function of cycle time is interesting. The performance increases in terms of throughput but delay increases because the Poisson traffic must wait until the end of the scheduled period which adds to its delay tremendously for the 10,000 slot time cycle.

Plots of delay versus throughput, throughput versus offered load, collisions versus offered load, delay versus offered load, packets lost versus offered load are pre sented in Figures 6.1 through 6.33. It is important to note that offered load is generated load while throughput is a measure of packets that get through the channel. Packets that are lost due to insufficient queue space or excessive collisions are discarded.

Figures 6.1 and 6.2 indicate the output of the simulator for the Ethernet protocol with a pure Poisson load and the measured data from Gonsalves' paper.[GONS87] This was used to verify the accuracy(validity) of the simulator. Gonsalves' network was 1500 meters long and included a repeater while the simulator models a short network. This accounts for the slightly better performance of the simulated Ethernet.

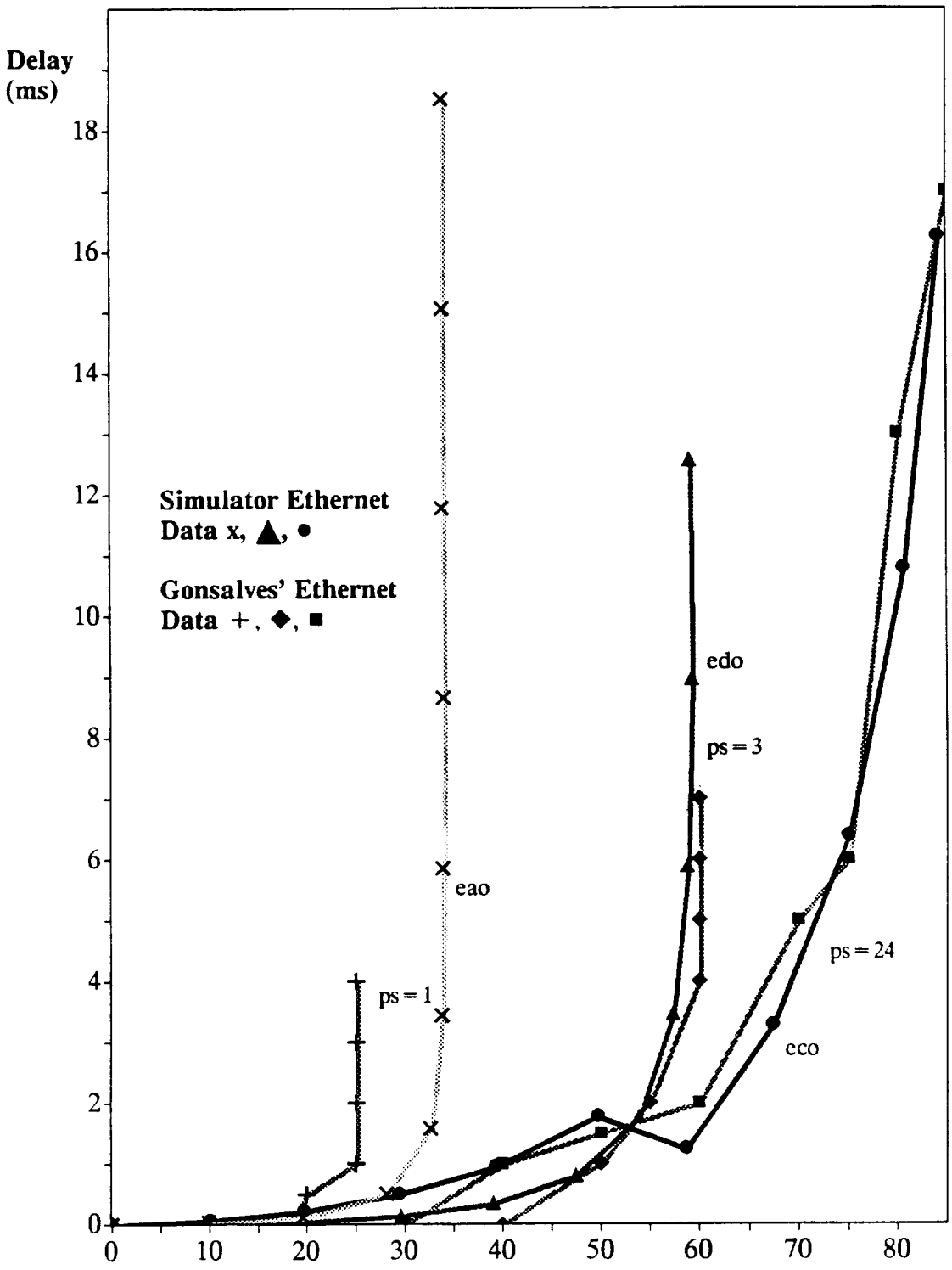


Figure 6.1 **Mean Delay in Milliseconds vs. Throughput for Various Packet Sizes. (1, 3 and 24 Slot Times)**

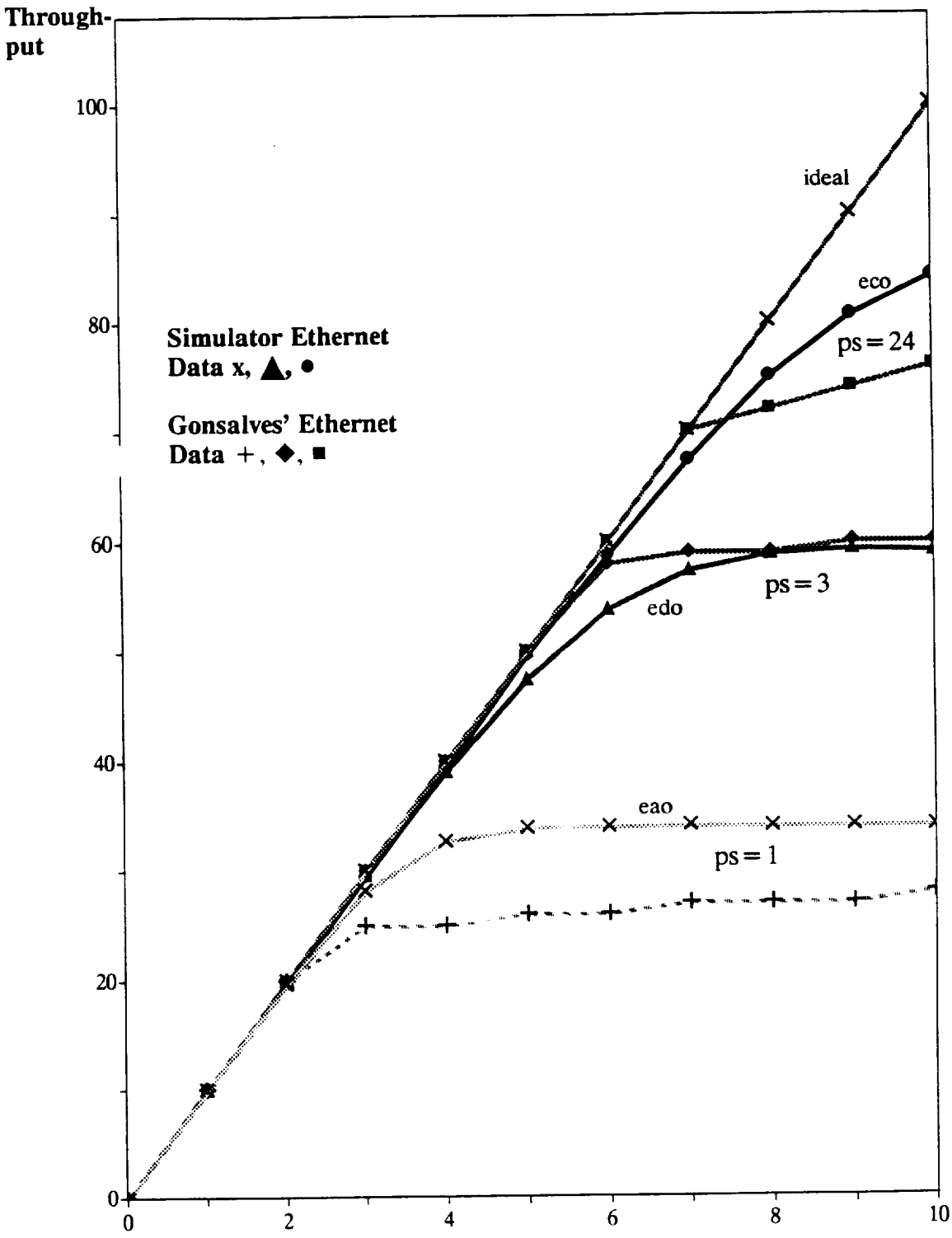


Figure 6.2 **Throughput vs. Offered Load in Megabits. (1, 3 and 24 Slot Times)**

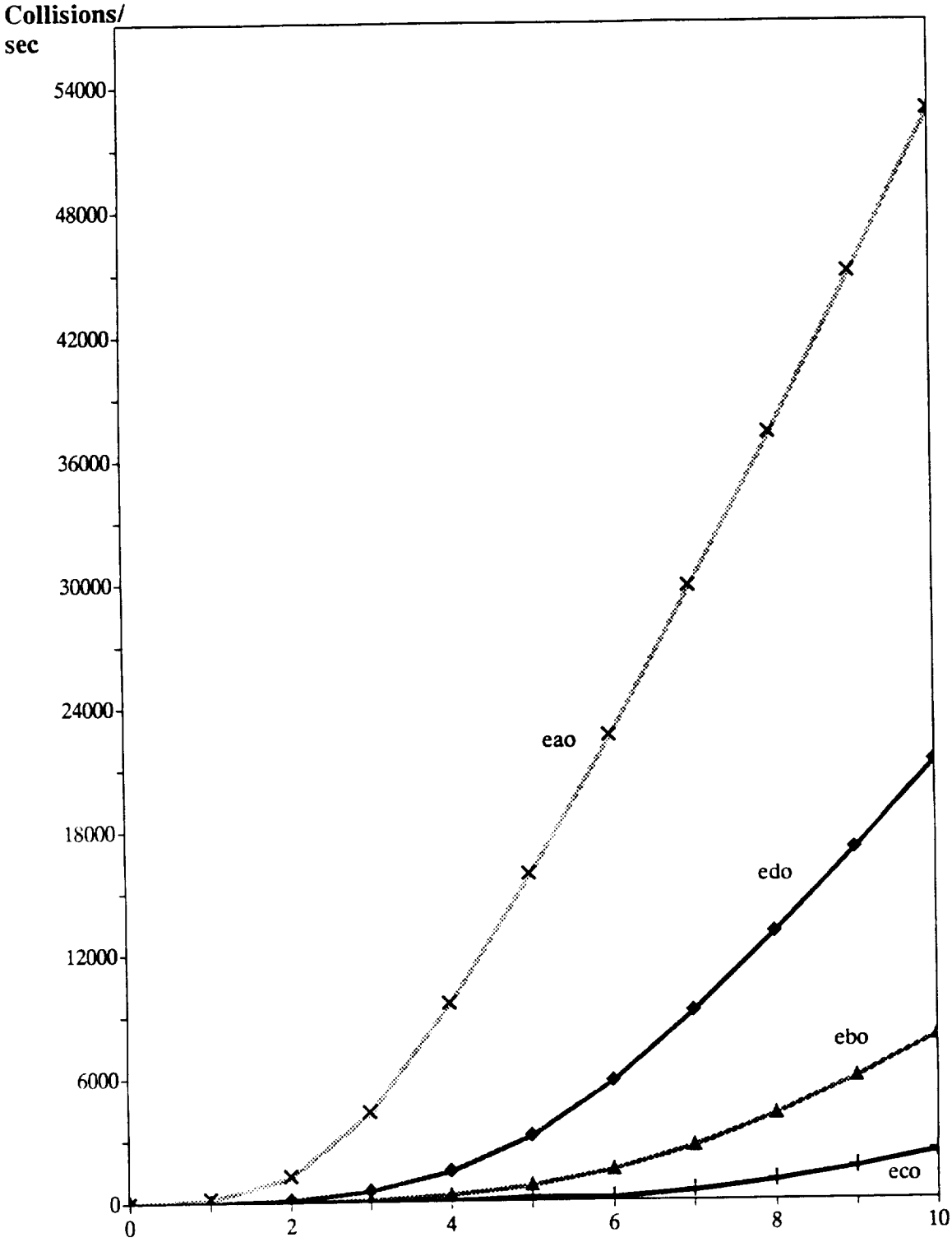


Figure 6.3 **Collisions/second vs. Offered Load in Megabits. (1, 3, 8 and 24 Slot Times)**

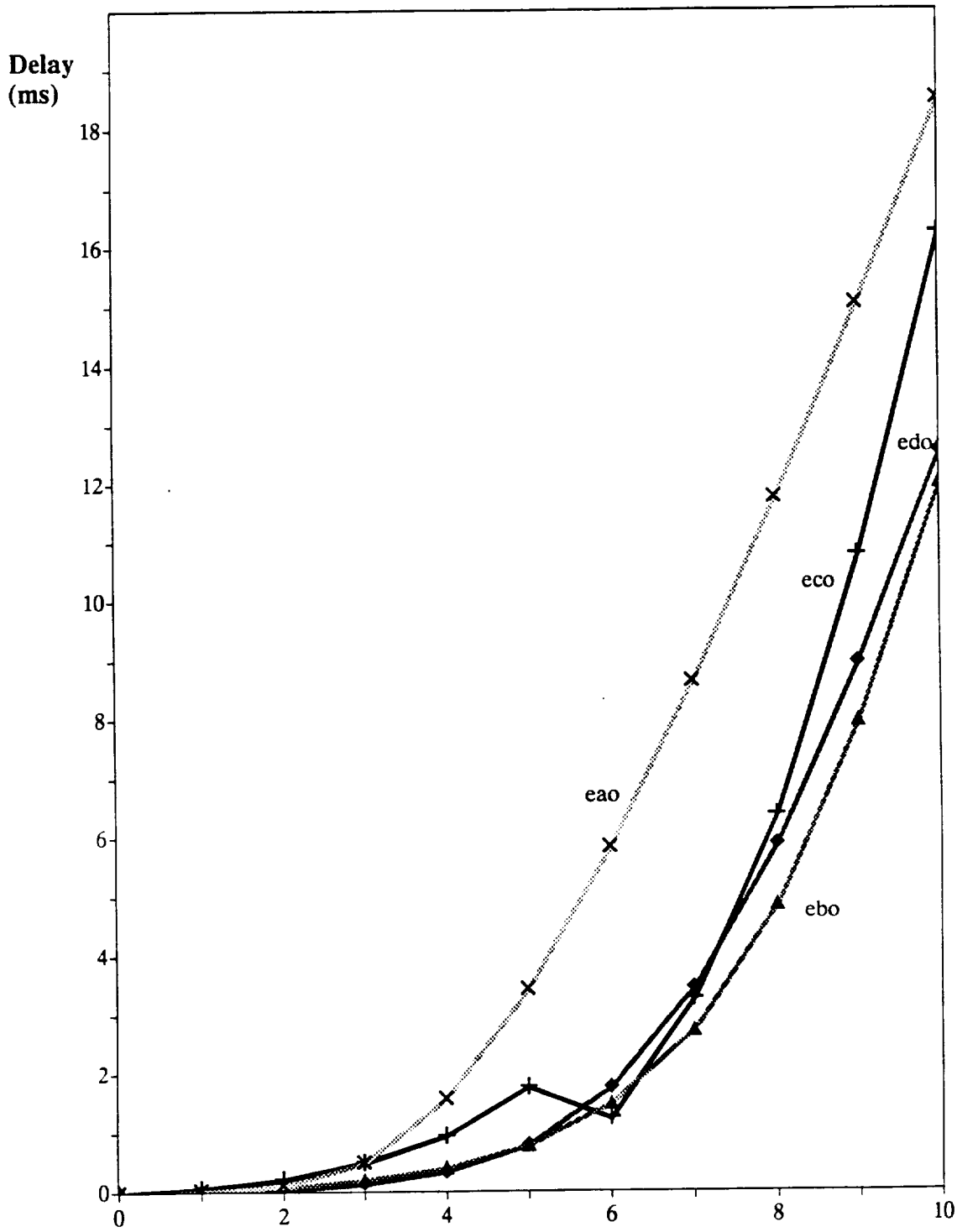


Figure 6.4 **Mean Delay in Milliseconds vs. Offered Load in Megabits for Various Packet Sizes. (1, 3, 8 and 24 Slot Times)**

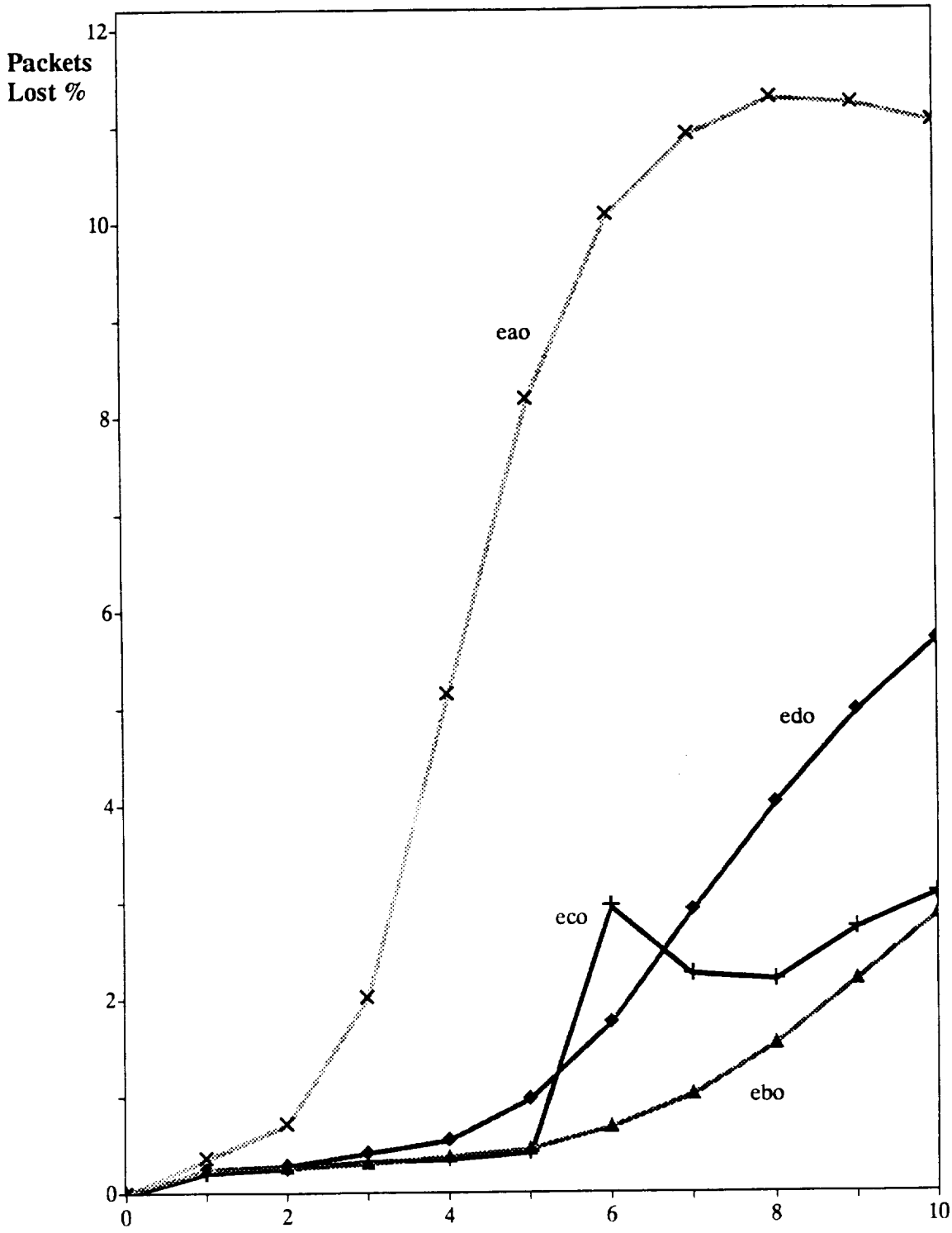


Figure 6.5 **Packets Lost vs. Offered Load in Megabits. (1, 3, 8 and 24 Slot Times)**

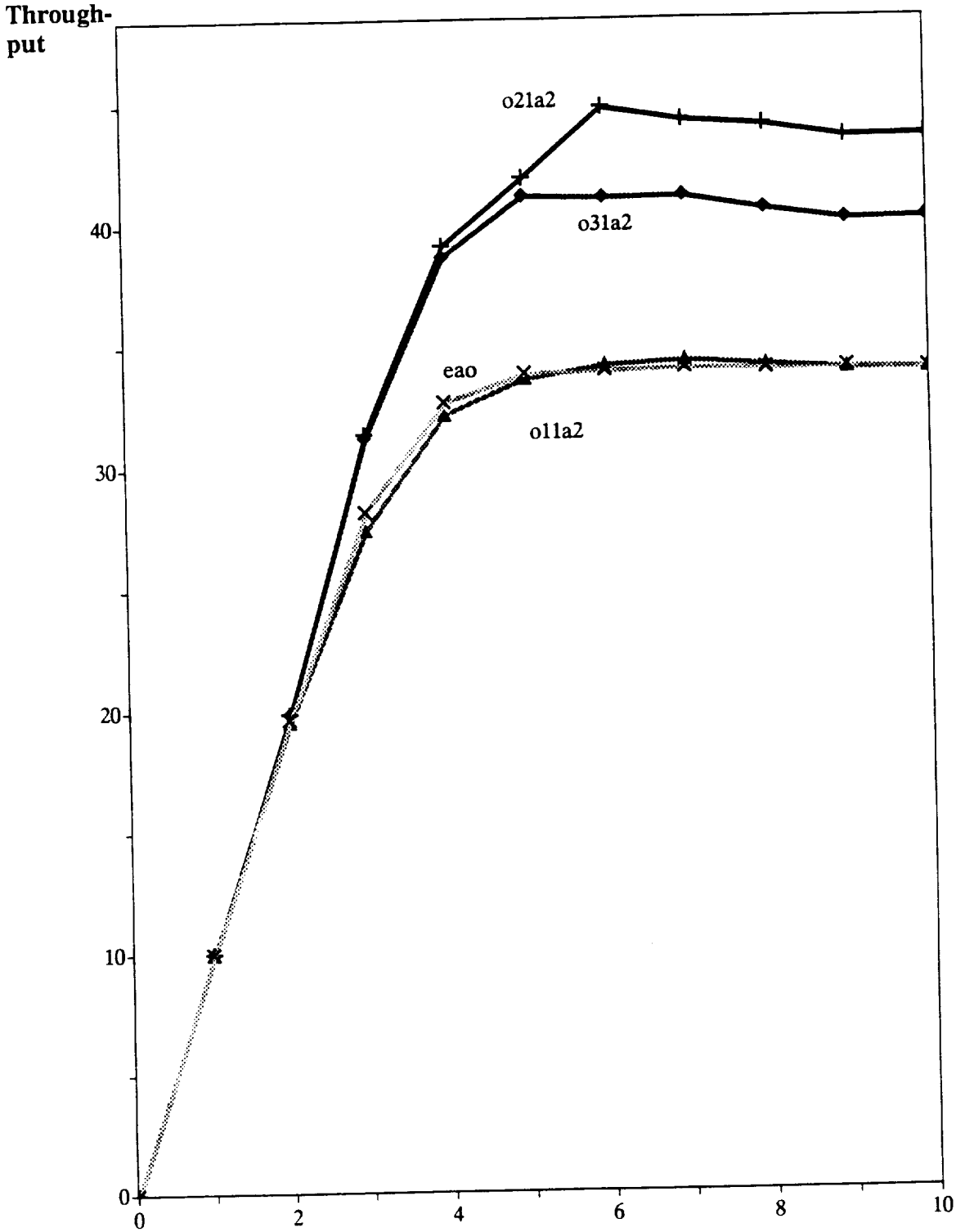


Figure 6.6 **Throughput vs. Offered Load for a Packet Size of 1 Slot Time (64 Bytes)**

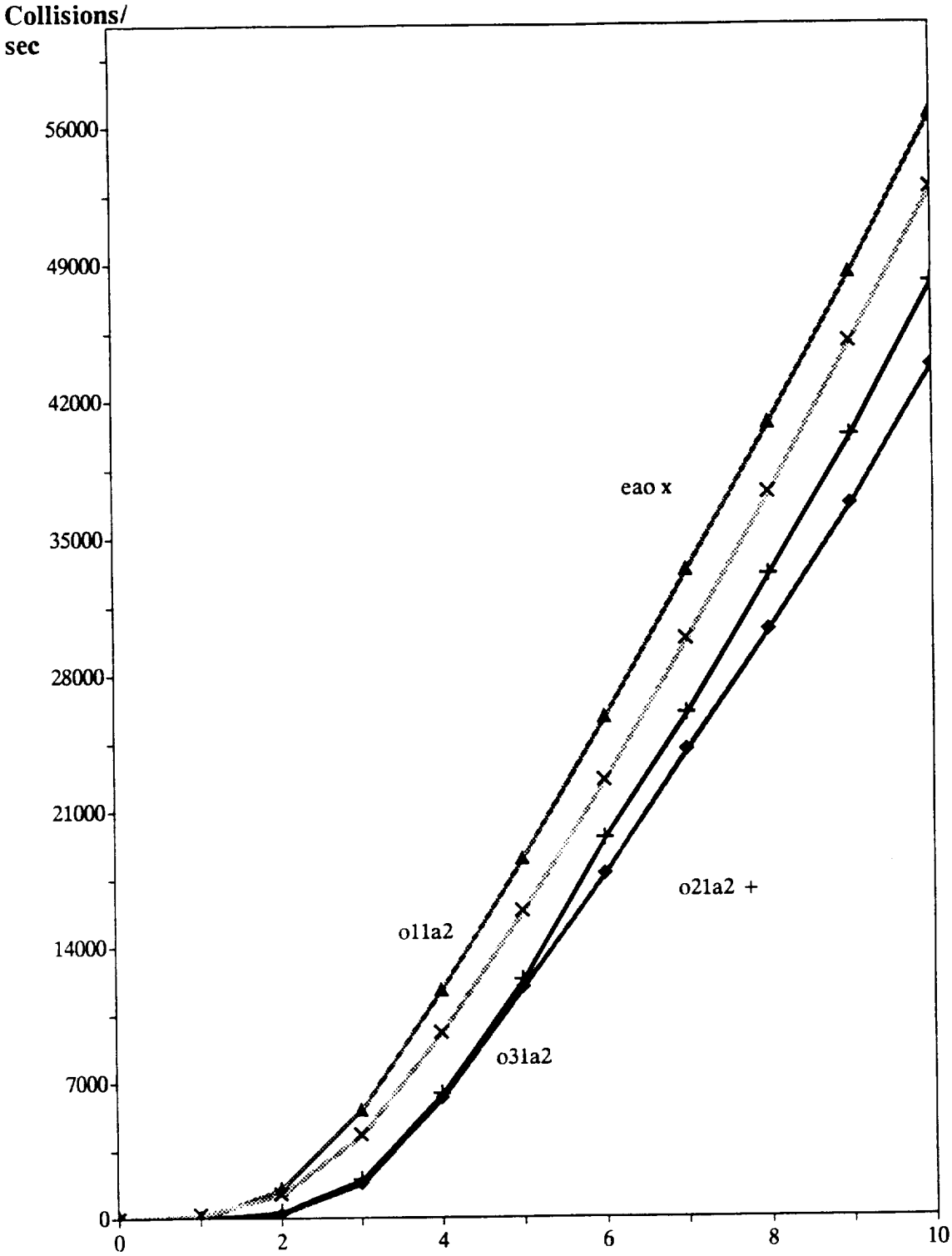


Figure 6.7 **Collisions/sec vs. Offered Load in Megabits for a Packet Length of 1 Slot Time**

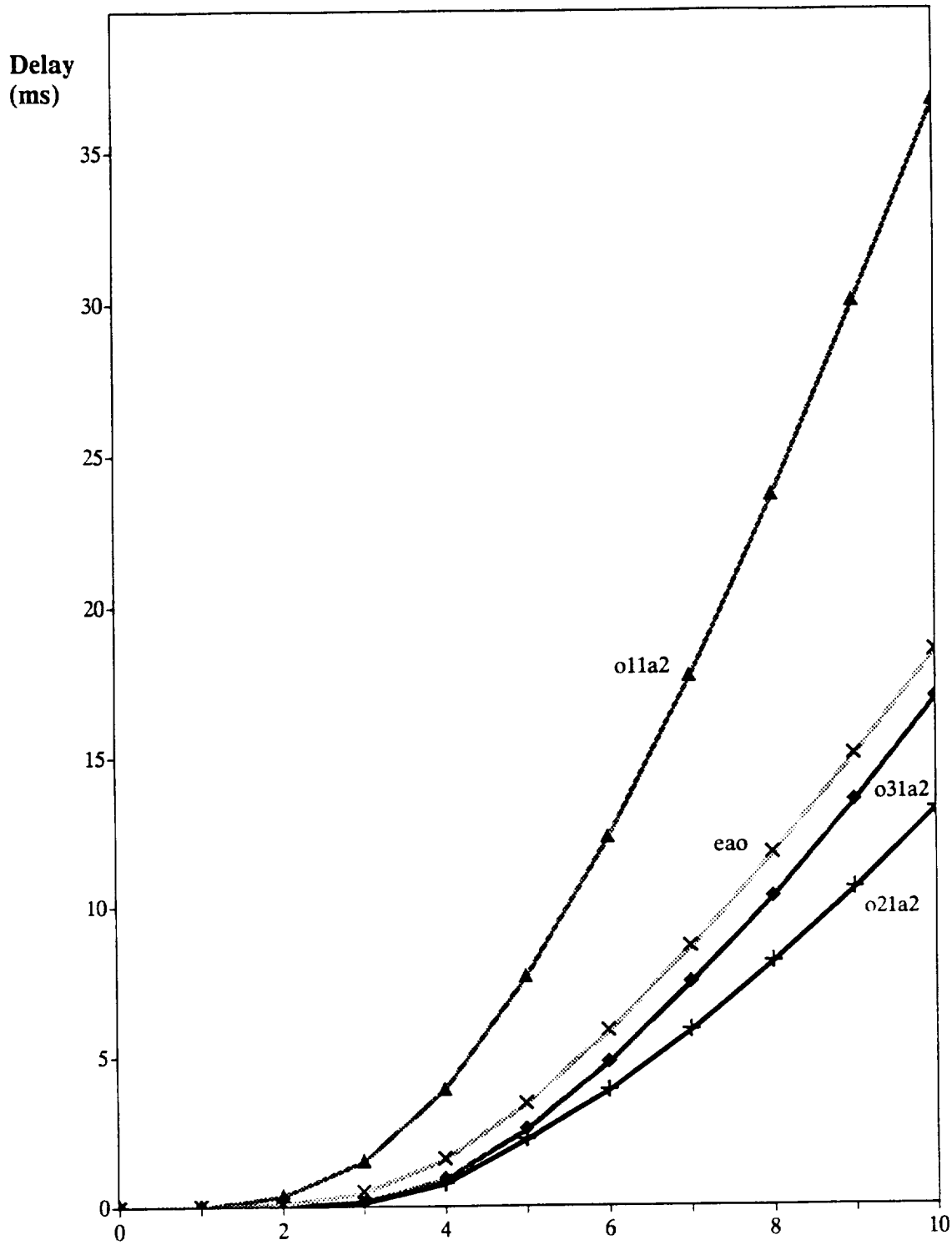


Figure 6.8 **Mean Delay in Milliseconds vs. Offered Load in Megabits for a Packet Length of 1 Slot Time**

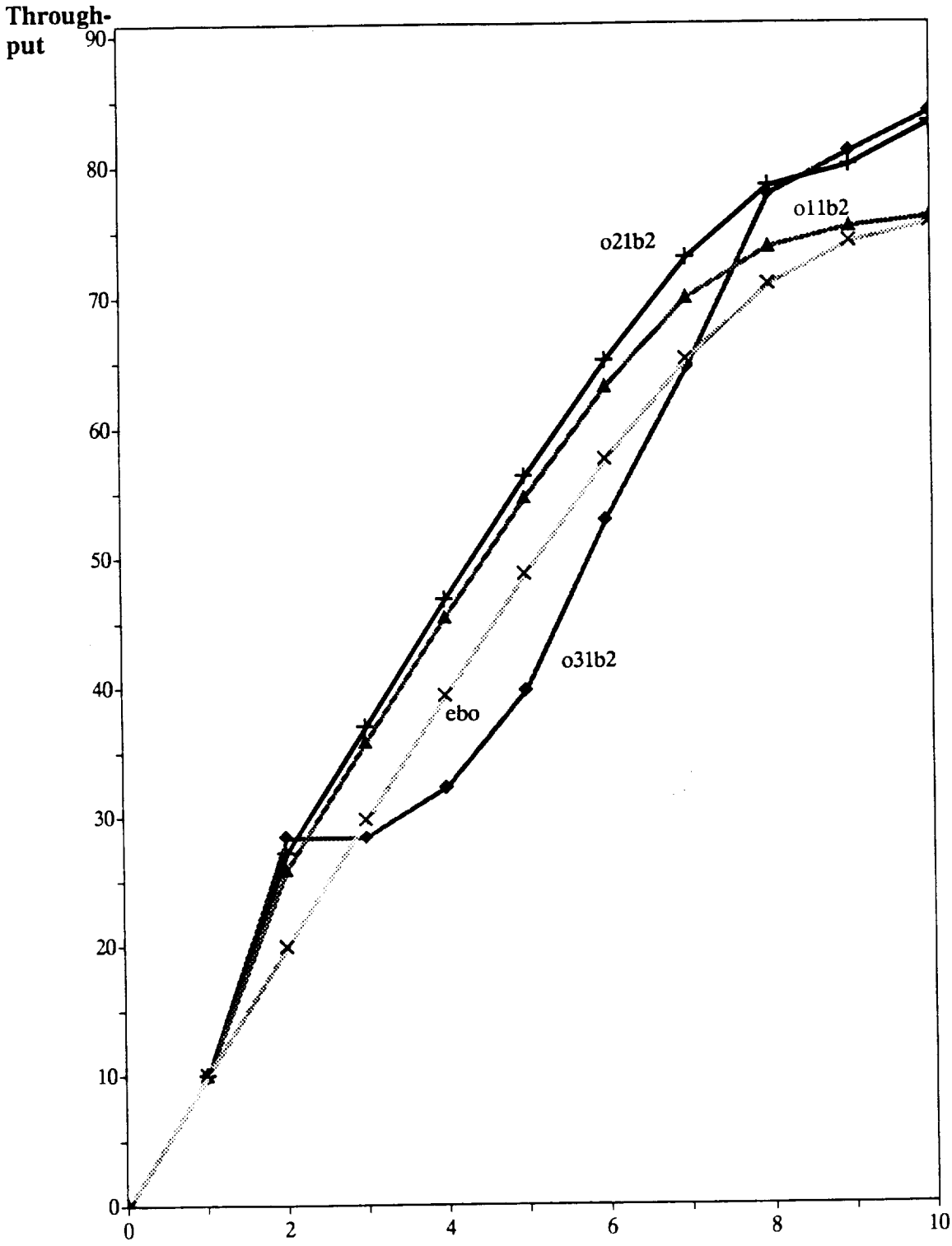


Figure 6.9 **Throughput vs. Offered Load in Megabits for a Packet Length of 8 Slot Times**

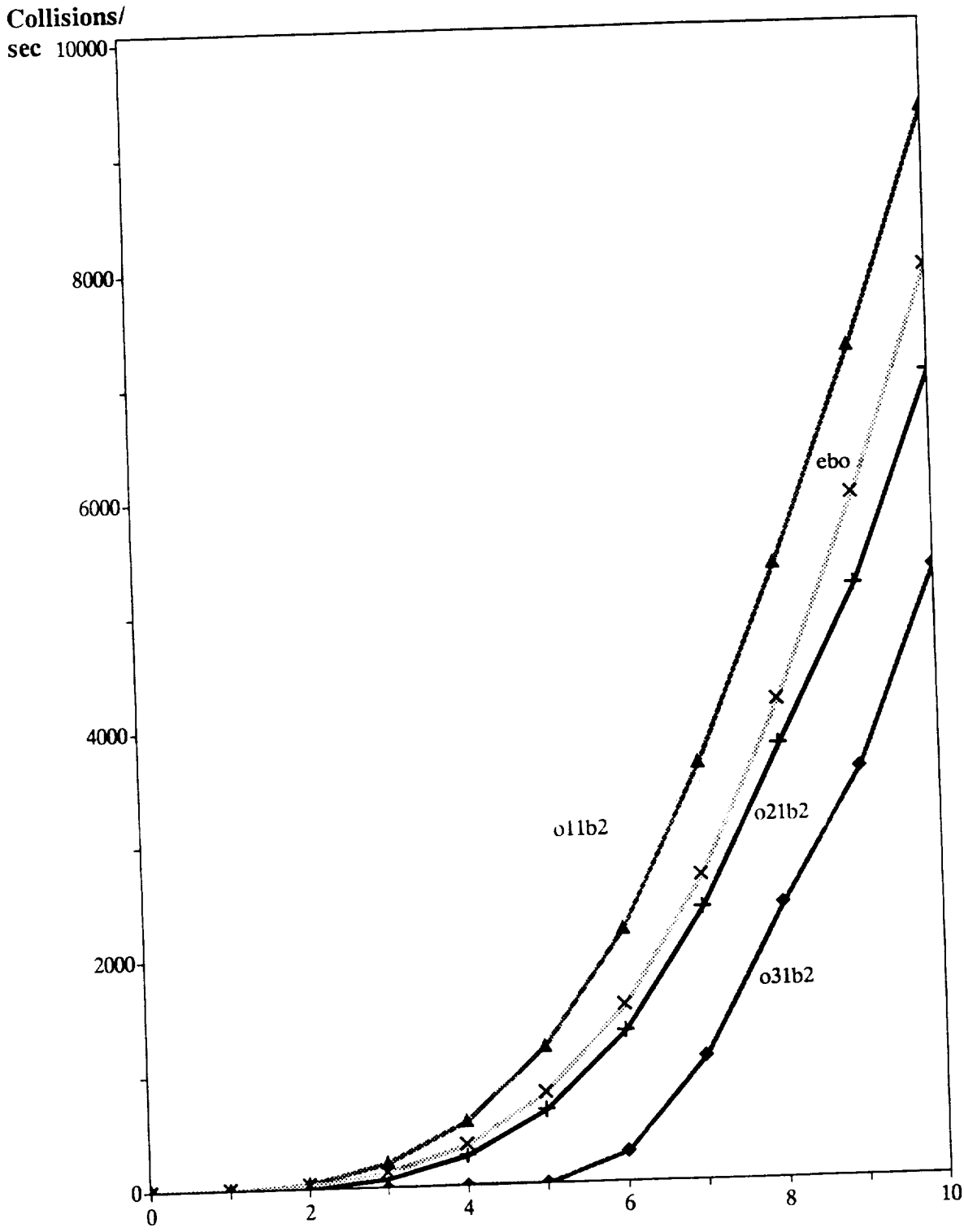


Figure 6.10 Collisions vs. Offered Load in Megabits for a Packet Length of 8 Slot Times

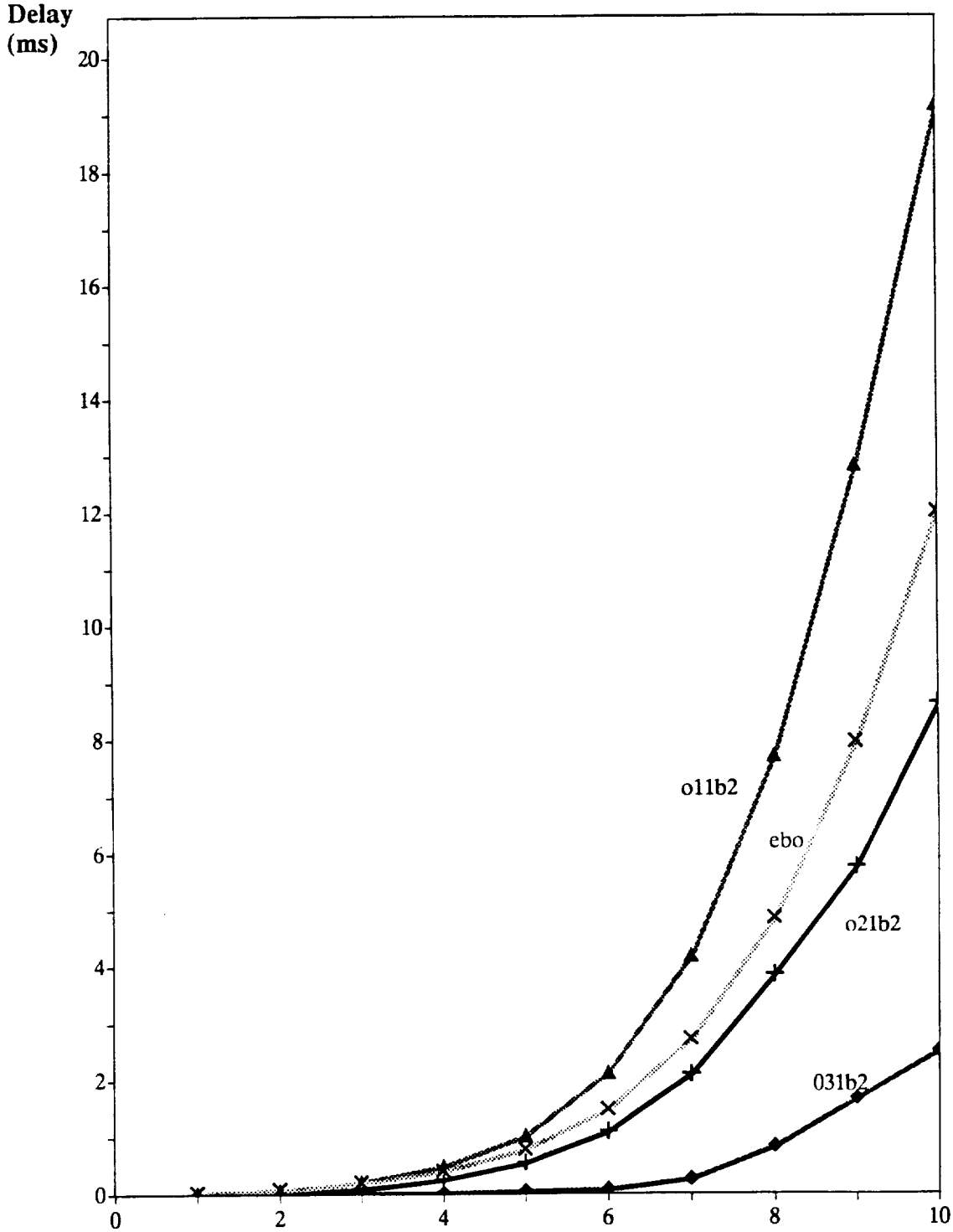


Figure 6.11 Mean Delay in Milliseconds vs. Offered Load in Megabits for a Packet Length of 8 Slot Times

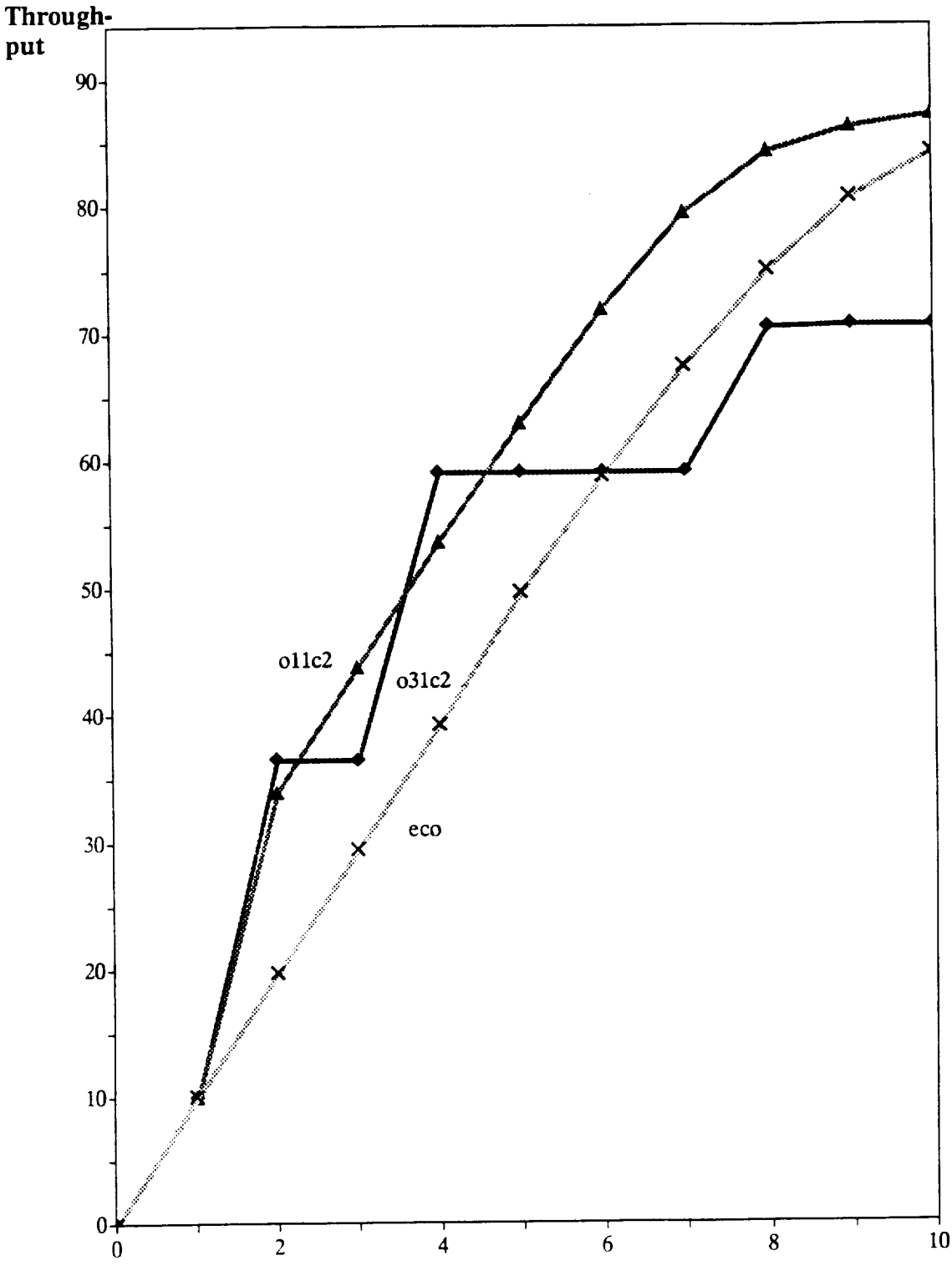


Figure 6.12 Throughput vs. Offered Load in Megabits for a Packet Length of 24 Slot Times

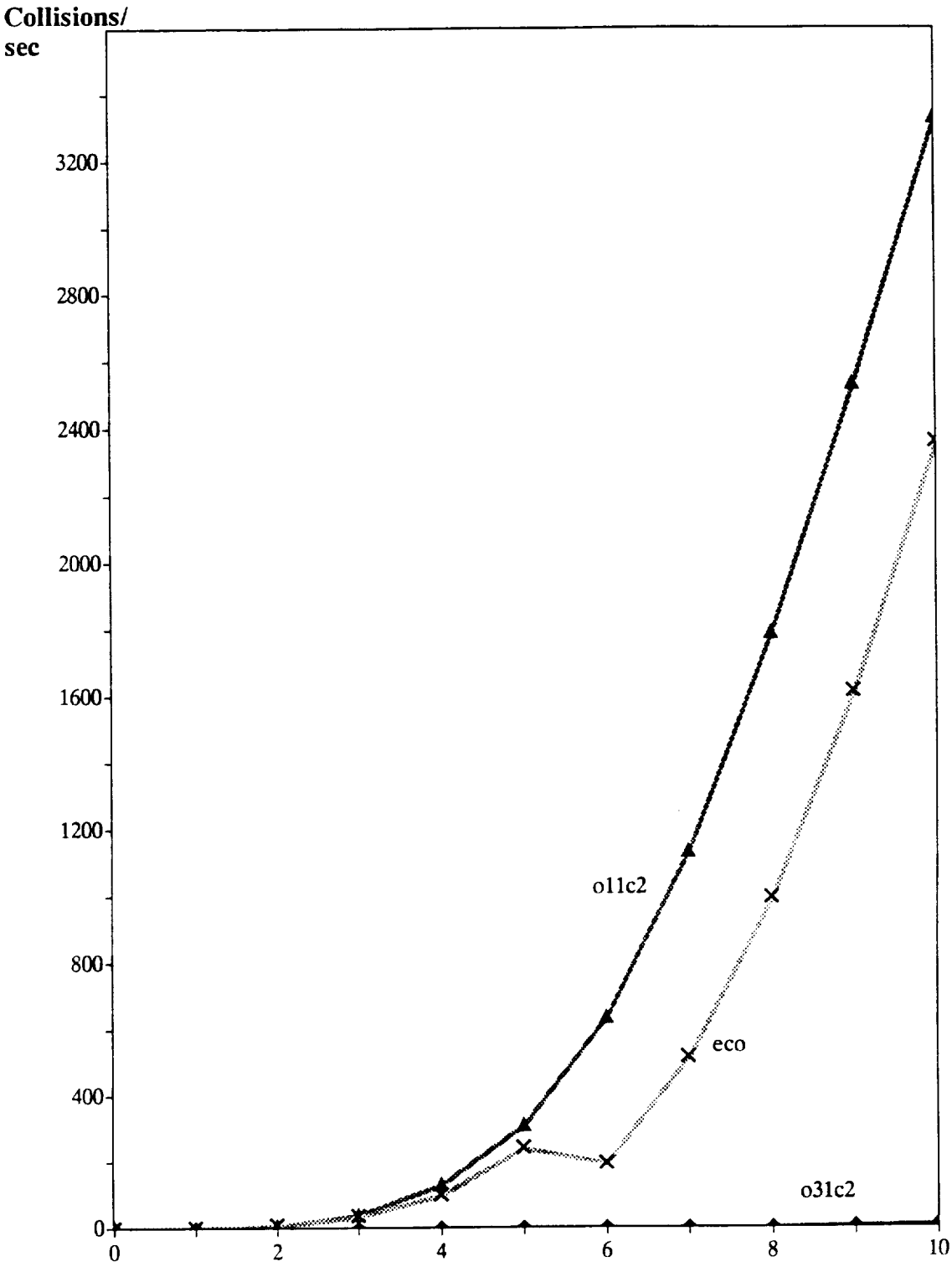


Figure 6.13 Collisions/sec vs. Offered Load in Megabits for a Packet Length of 24 Slot Times

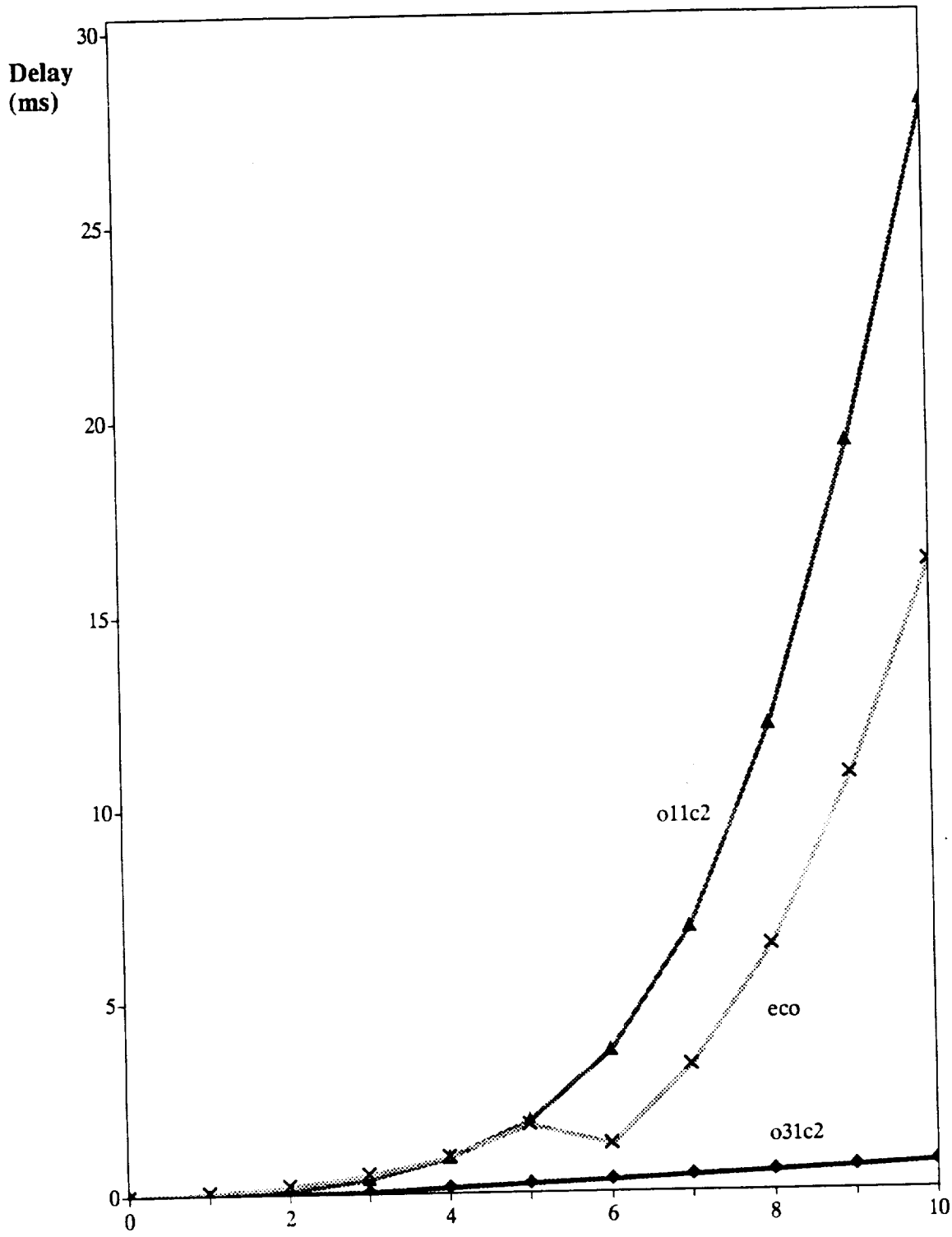


Figure 6.14 **Mean Delay in Milliseconds vs. Offered Load in Megabits for a Packet Length of 24 Slot Times**

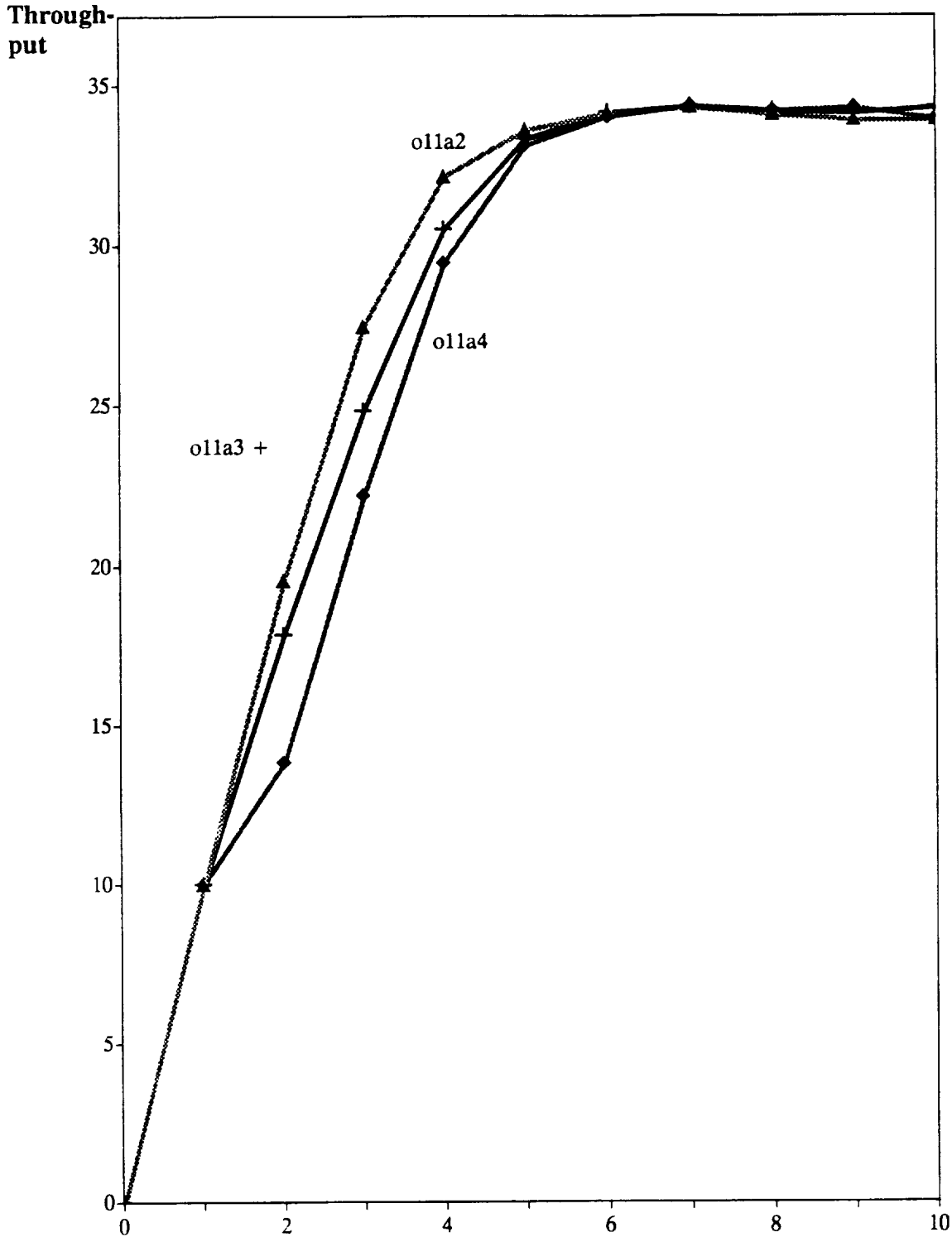


Figure 6.15 Throughput vs. Offered Load in Megabits for a Packet Length of 1 Slot Time

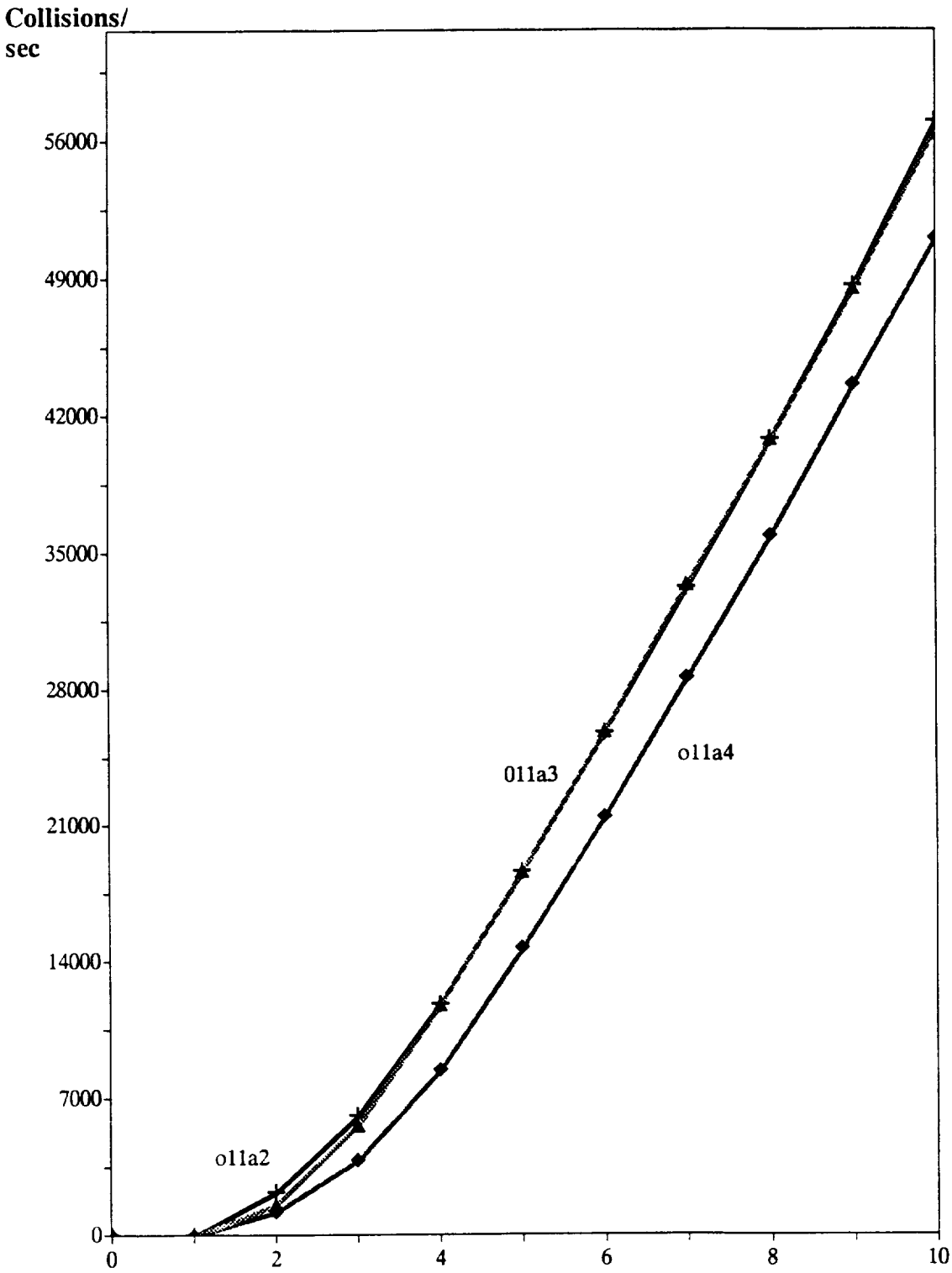


Figure 6.16 Collisions/sec vs. Offered Load in Megabits for a Packet Length of 1 Slot Time

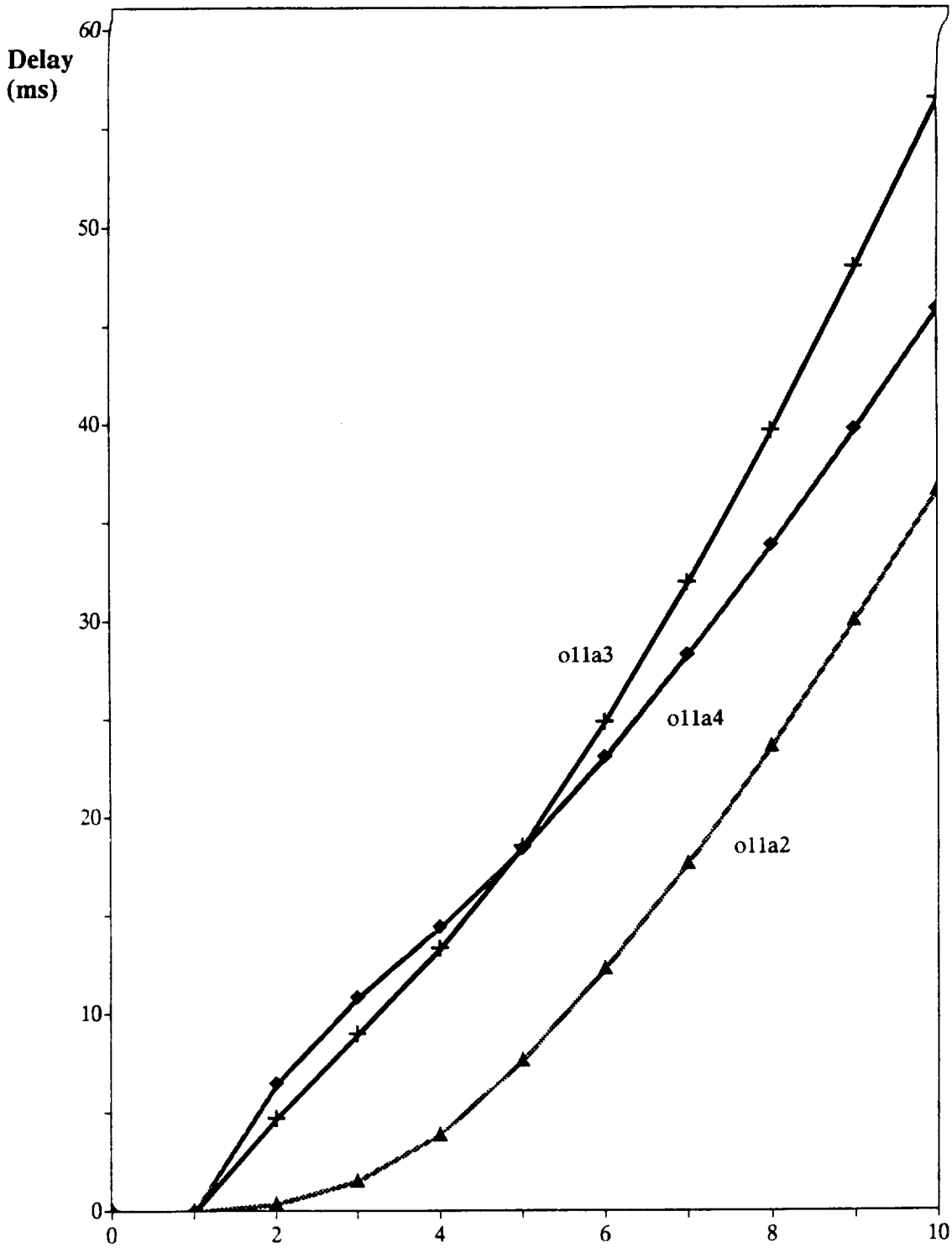


Figure 6.17 Mean Delay in Milliseconds vs. Offered Load in Megabits for a Packet Length of 1 Slot Time

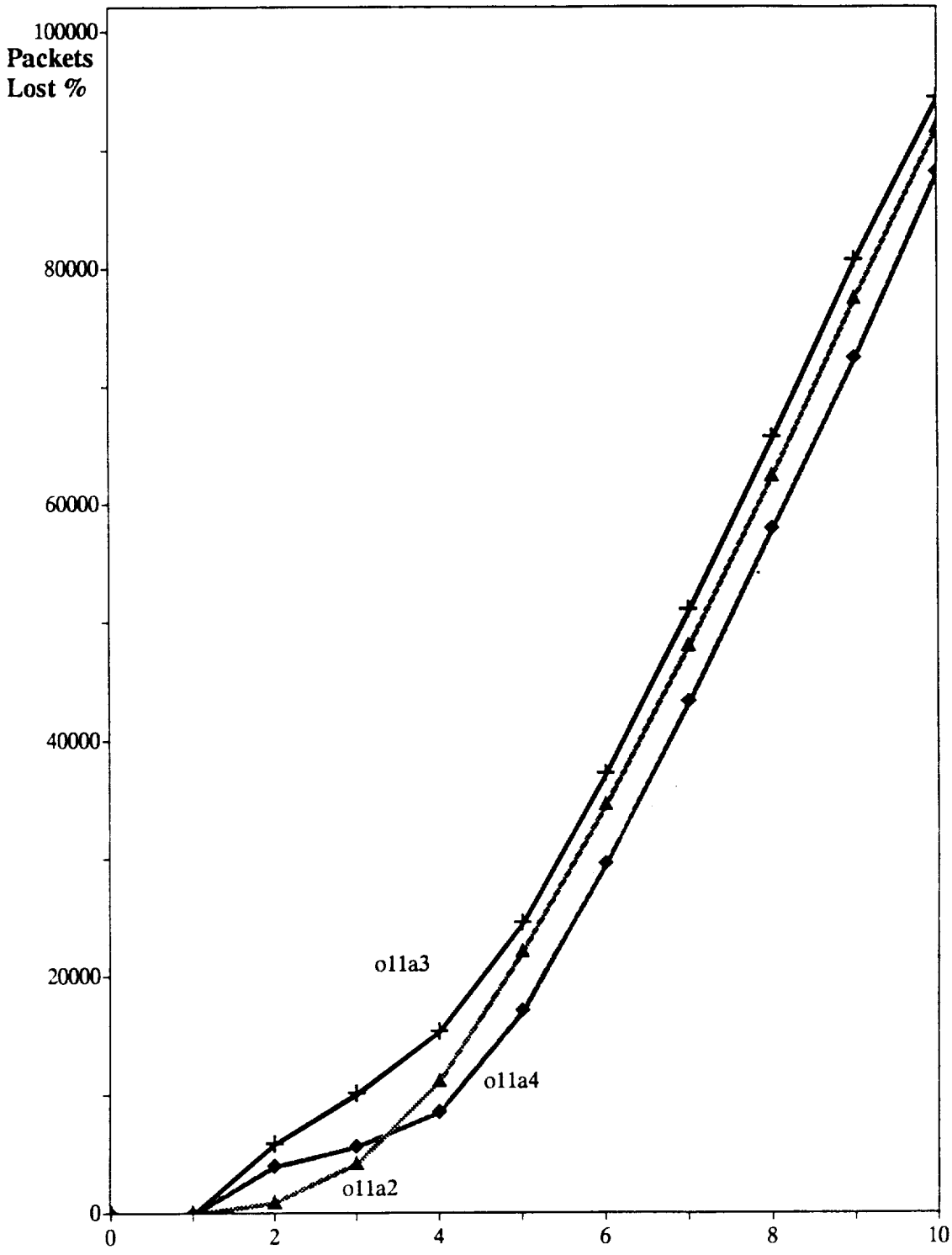


Figure 6.18 Packets Lost vs. Offered Load in Megabits for a Packet Length of 1 Slot Time

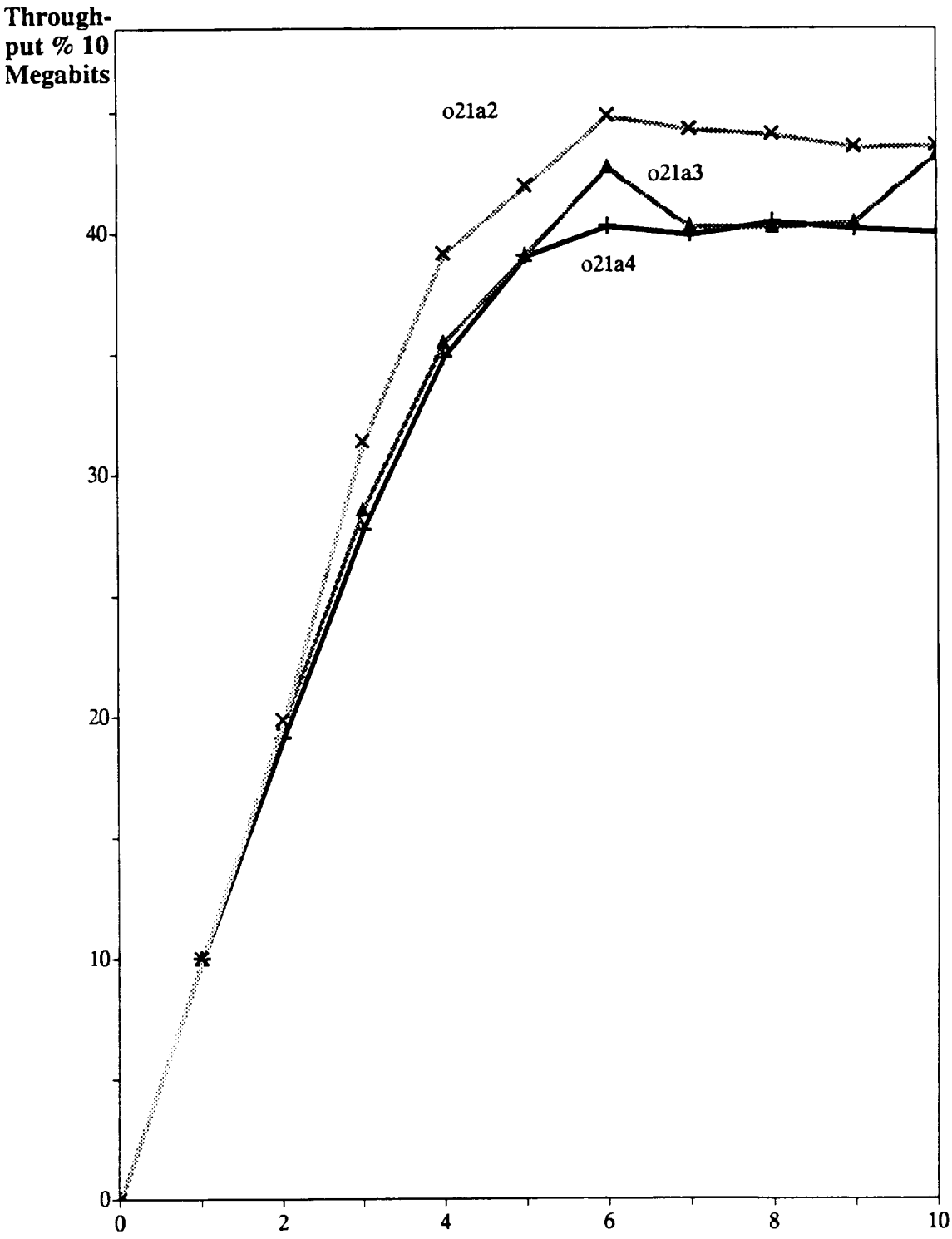


Figure 6.19 Throughput vs. Offered Load in Megabits for a Packet Length of 1 Slot Time for Various Cycle Lengths

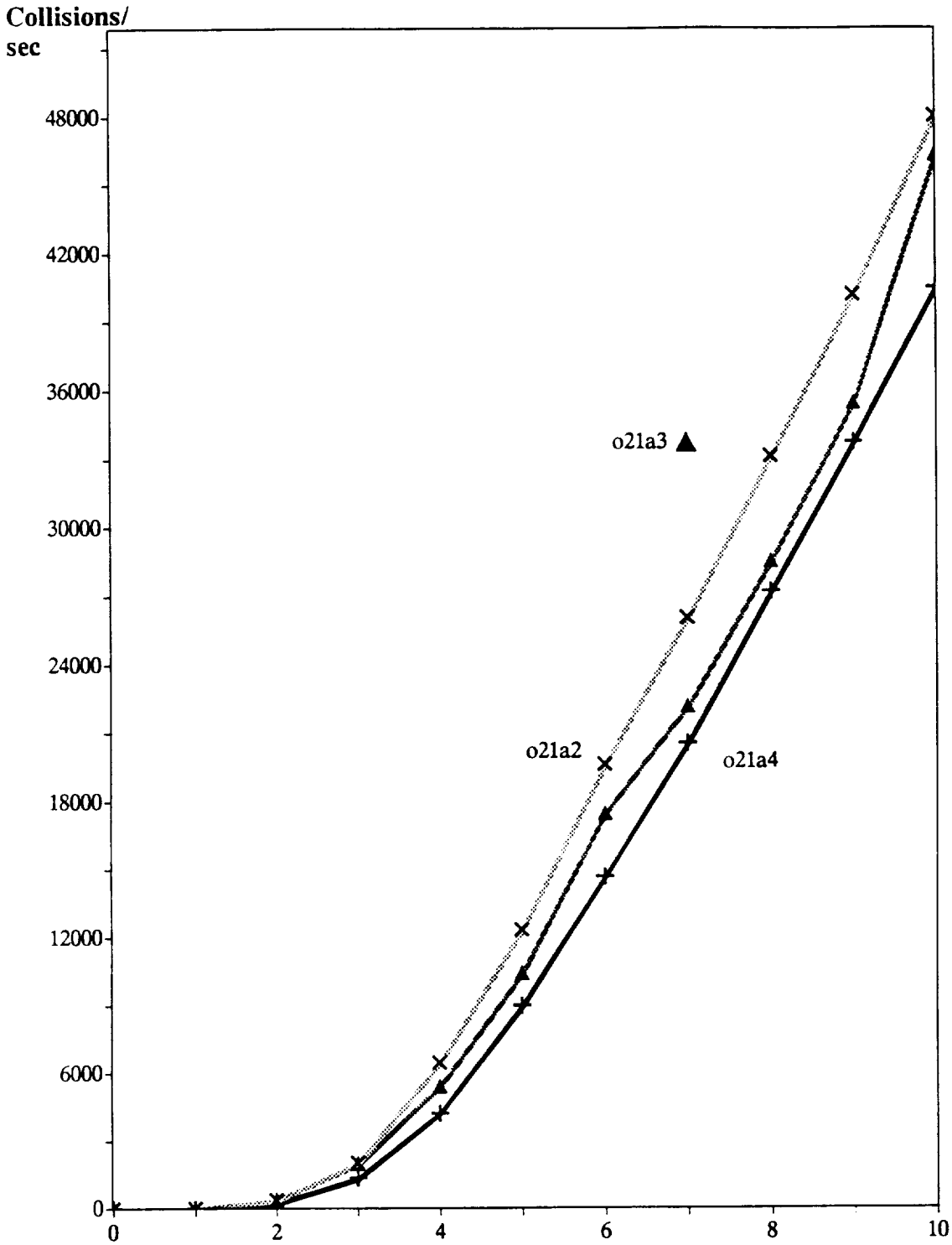


Figure 6.20 Collisions/sec vs. Offered Load in Megabits for a Packet Length of 1 Slot Time for Different Cycle Lengths

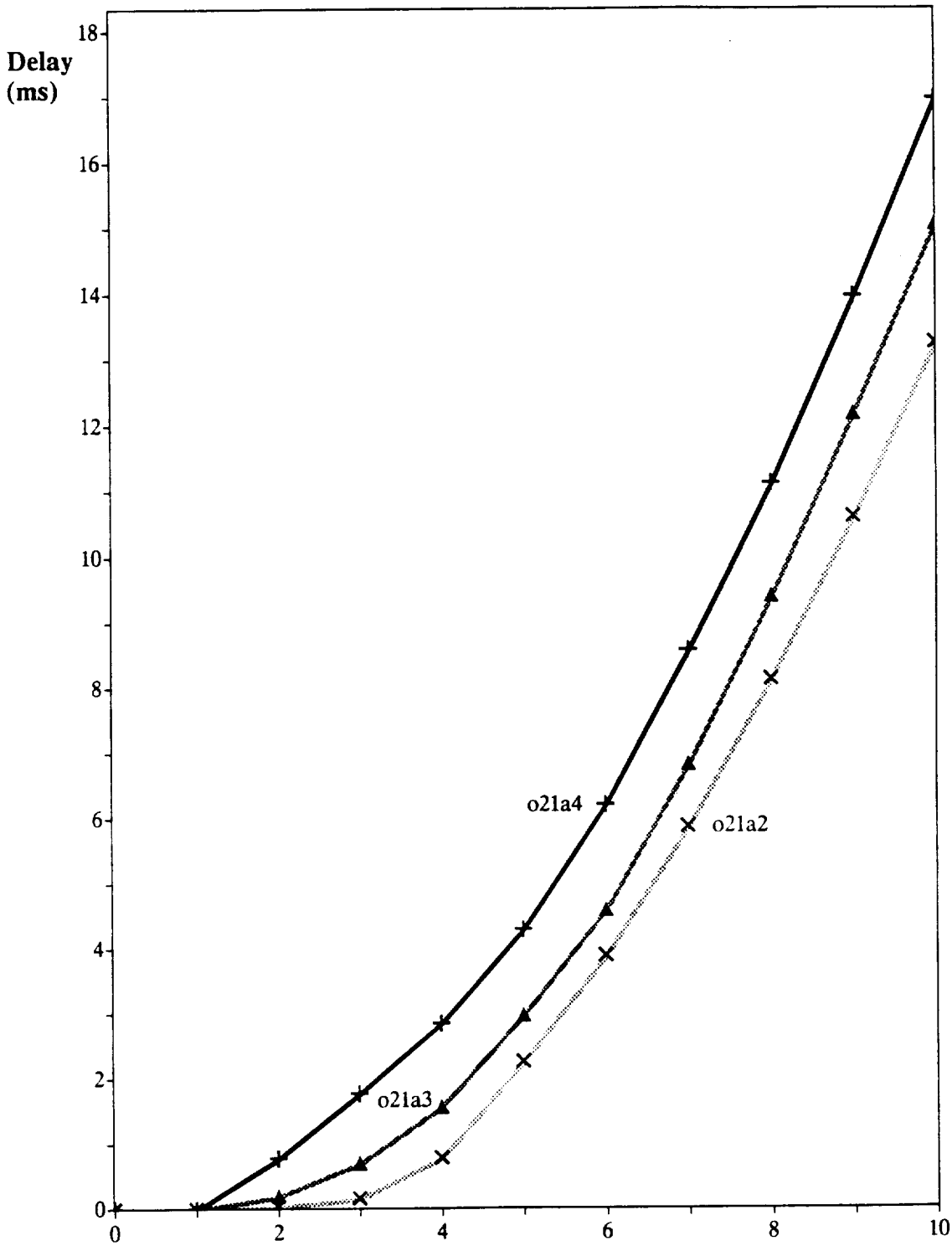


Figure 6.21 Mean Delay in Milliseconds vs. Offered Load in Megabits for a Packet Length of 1 Slot Time

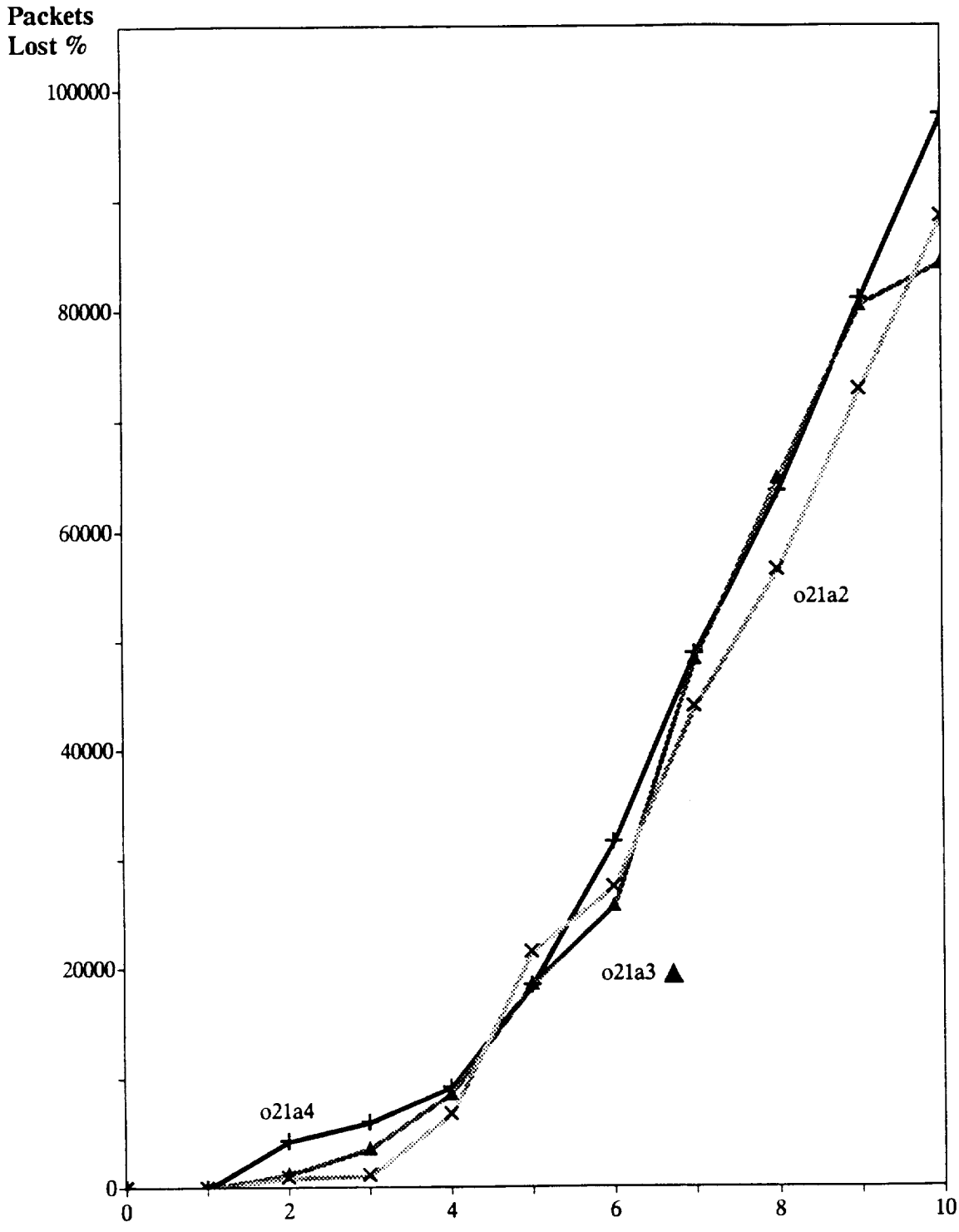


Figure 6.22 **Packets Lost vs. Offered Load in Megabits for a Packet Length of 1 Slot Time for Various Cycle Lengths**

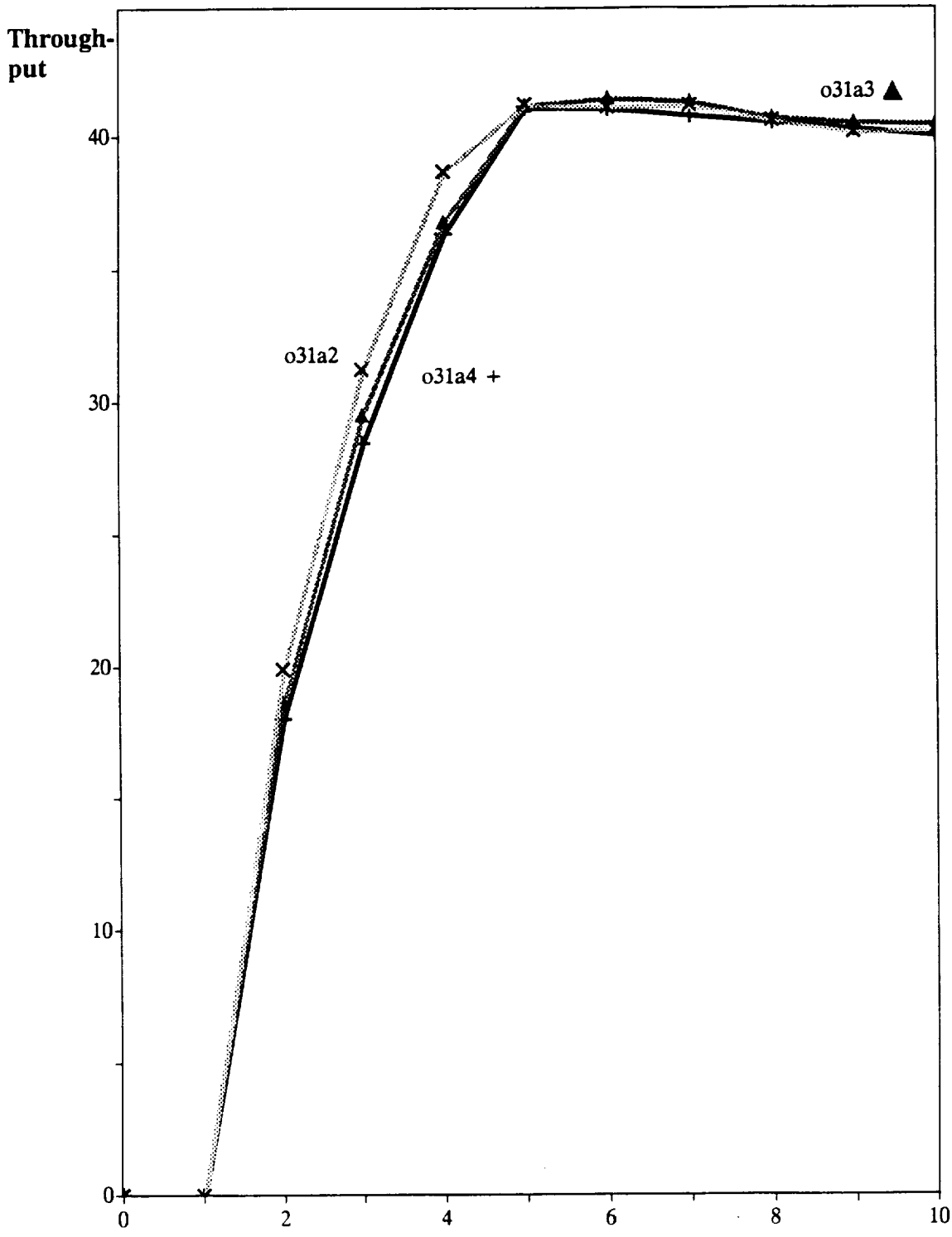


Figure 6.23 Throughput vs. Offered Load in Megabits for a Packet Length of 1 Slot Time

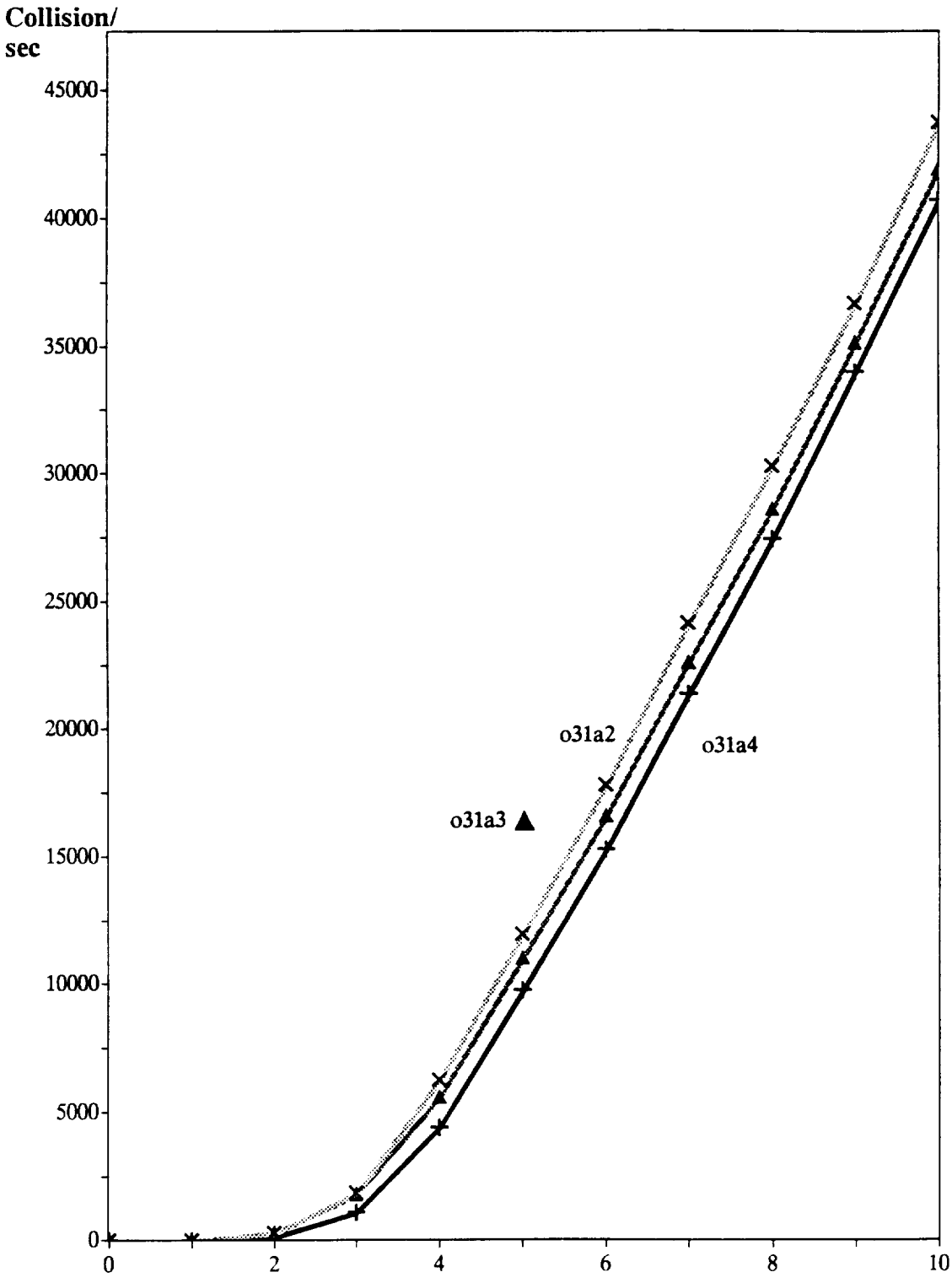


Figure 6.24 Collisions/second vs. Offered Load Megabits for a Packet Length of 1 Slot Time

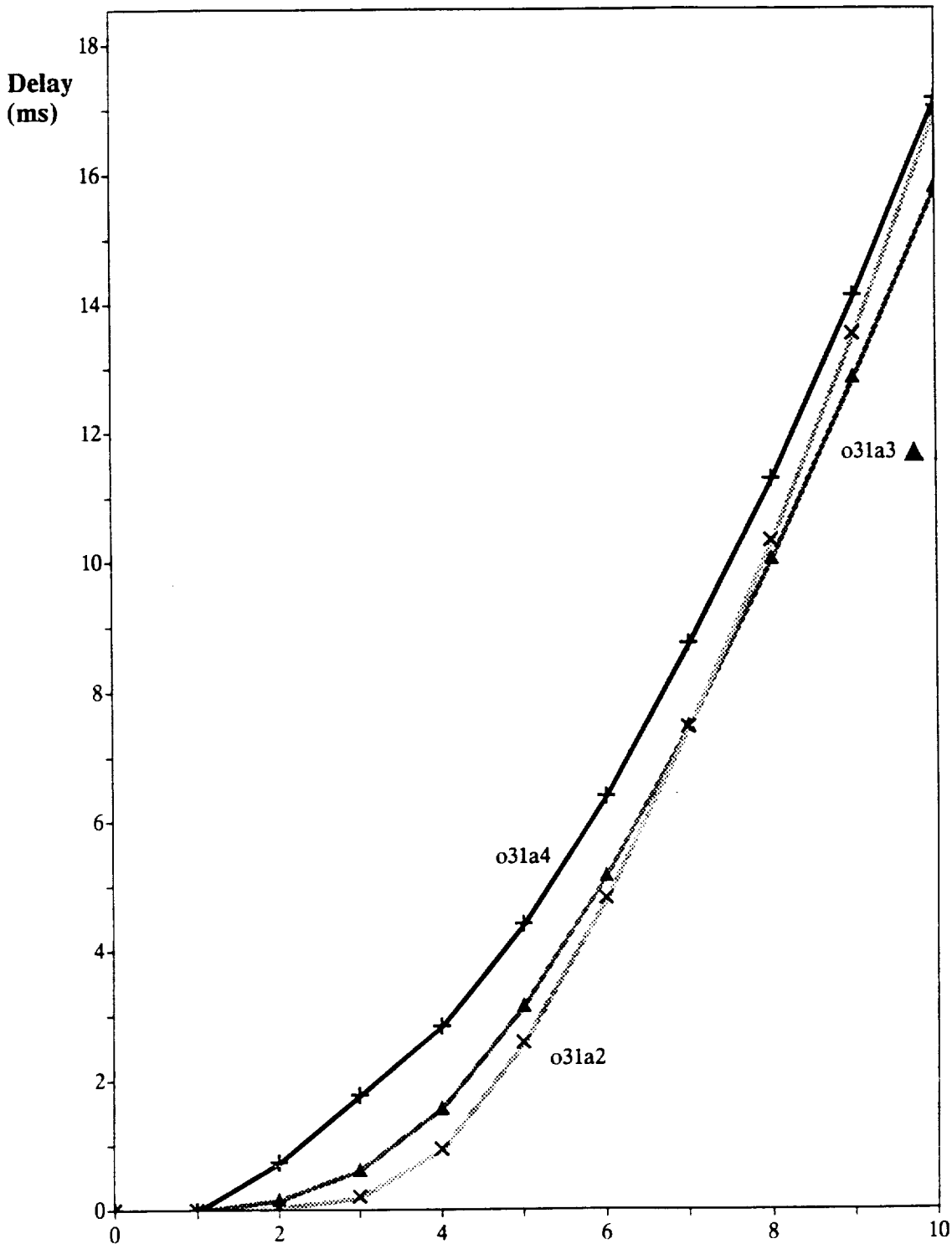


Figure 6.25 Mean Delay in Milliseconds vs. Offered Load in Megabits for a Packet Length of 1 Slot Time

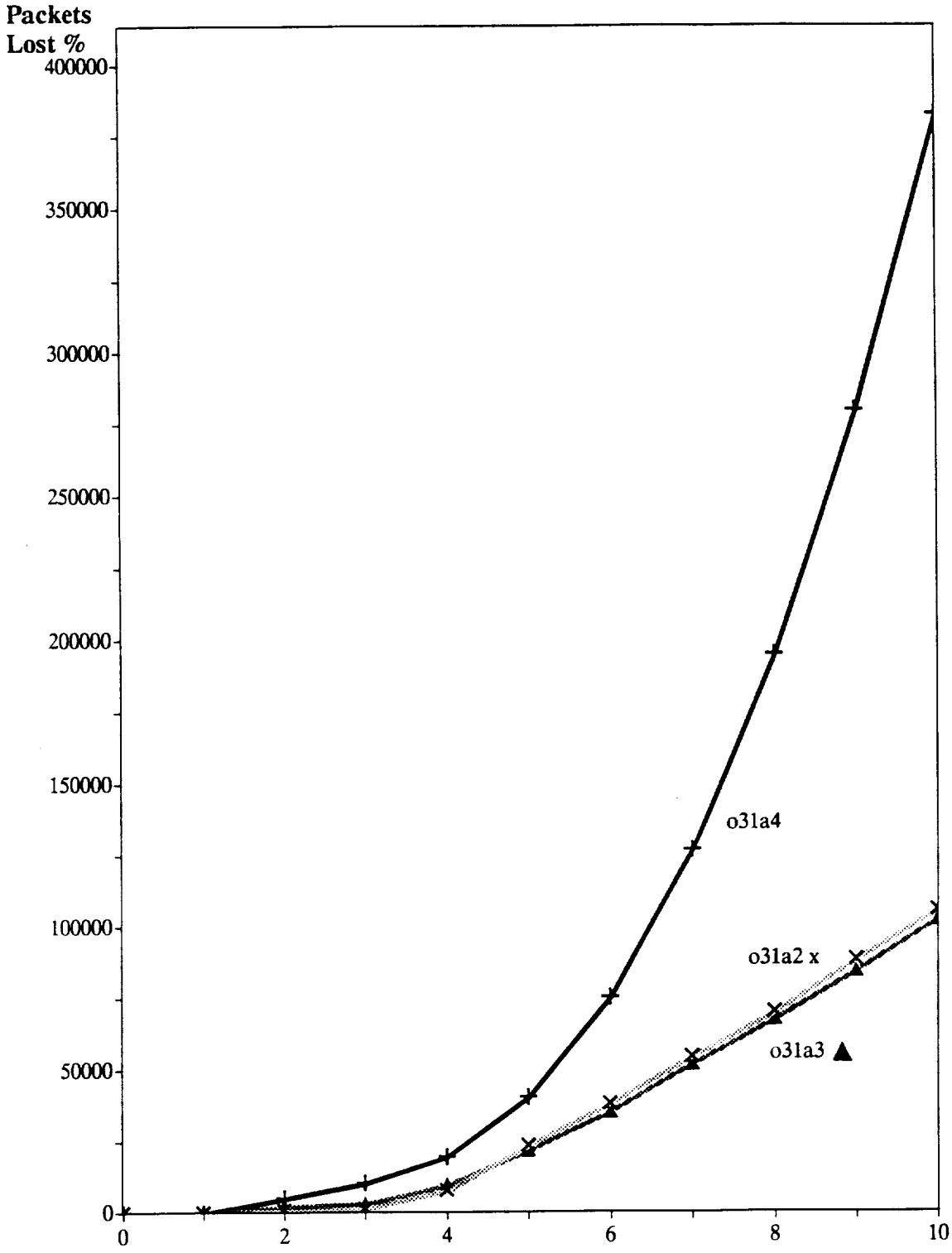


Figure 6.26 Packets Lost vs. Offered Load in Megabits for a Packet Length of 1 Slot Time

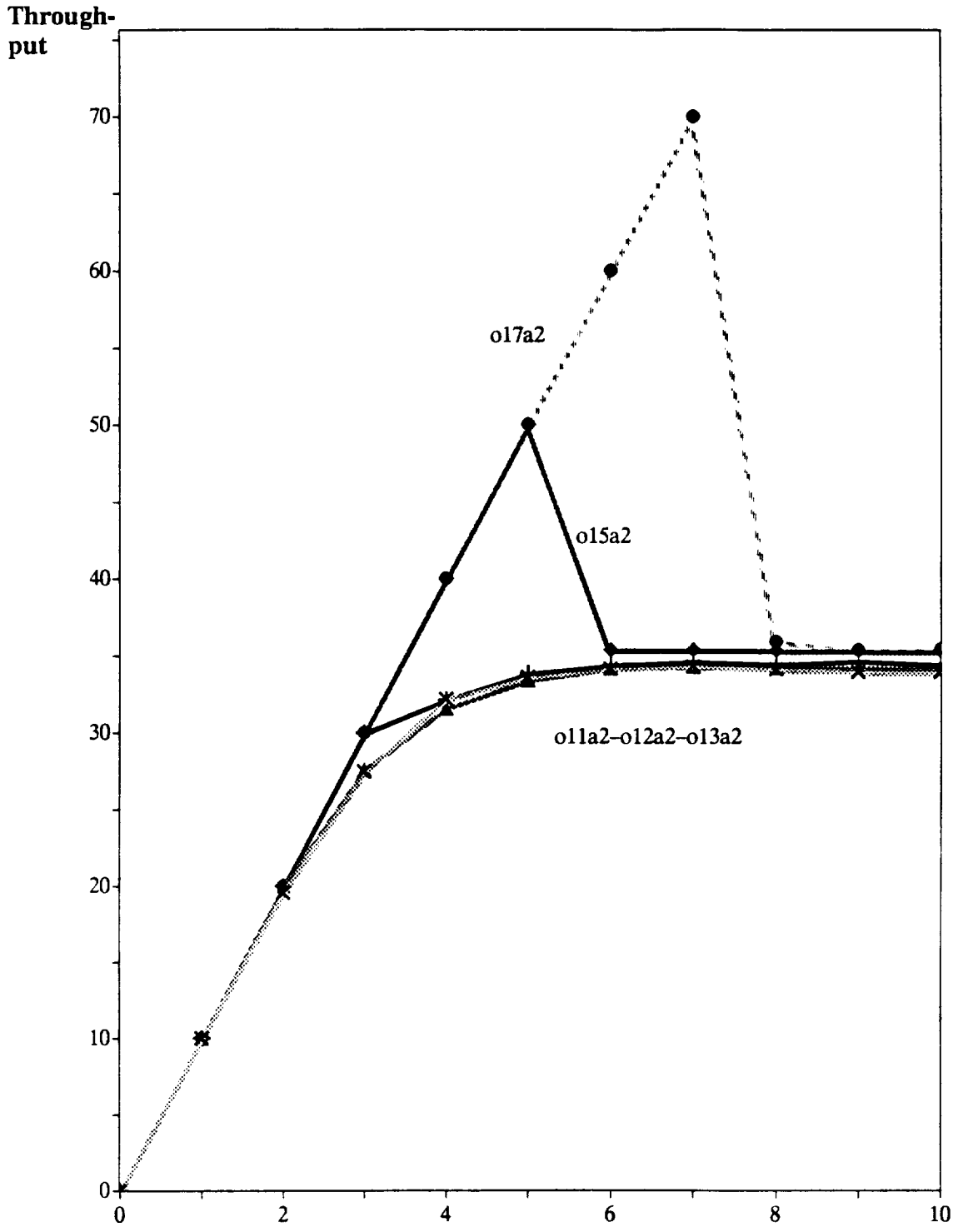


Figure 6.27 **Throughput vs. Offered Load in Megabits for a Packet Length of 1 Slot Time for Scheduled Traffic on Ethernet**

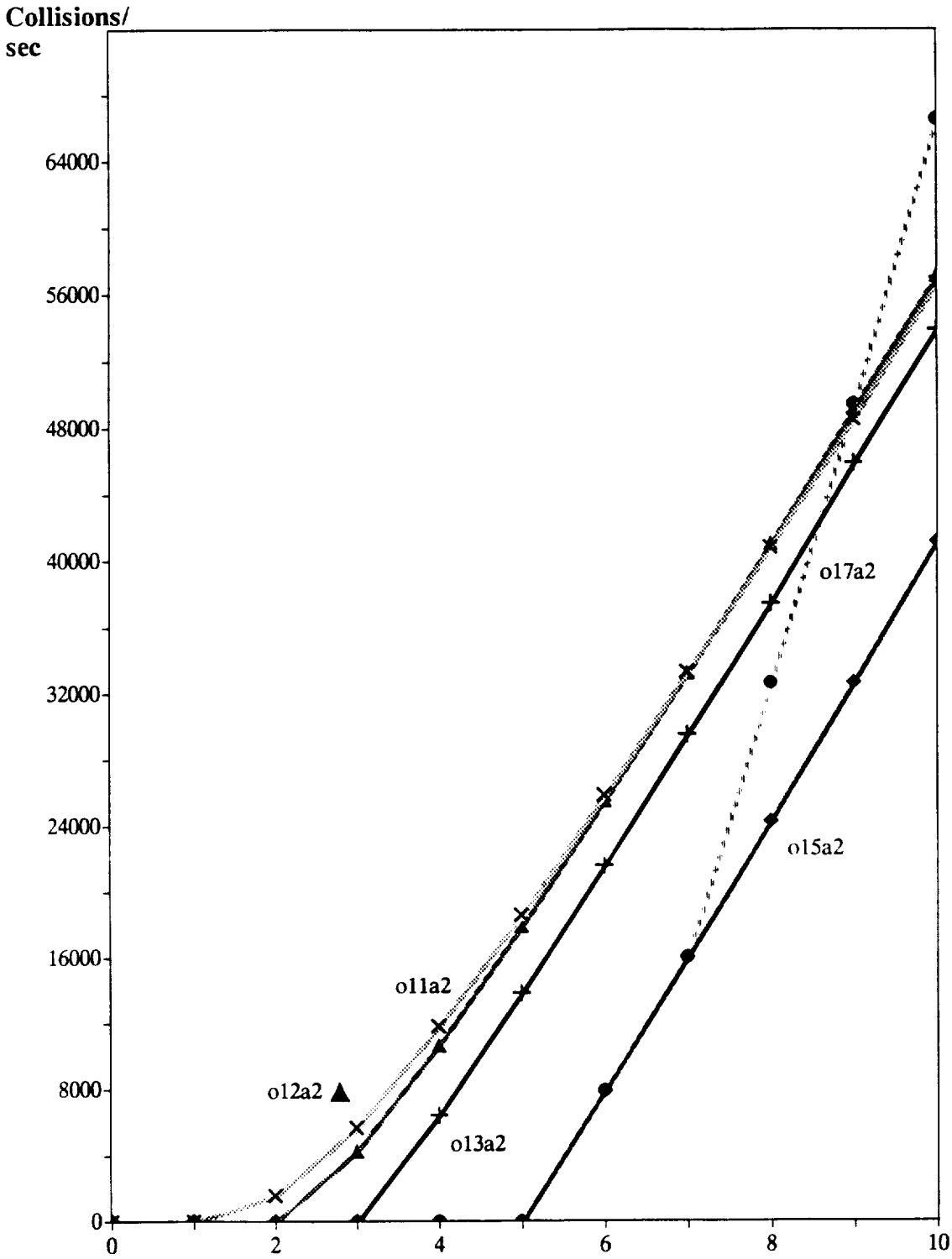


Figure 6.28 Collisions/second vs. Offered Load in Megabits for a Packet Length of 1 Slot Time for Scheduled Traffic on Ethernet

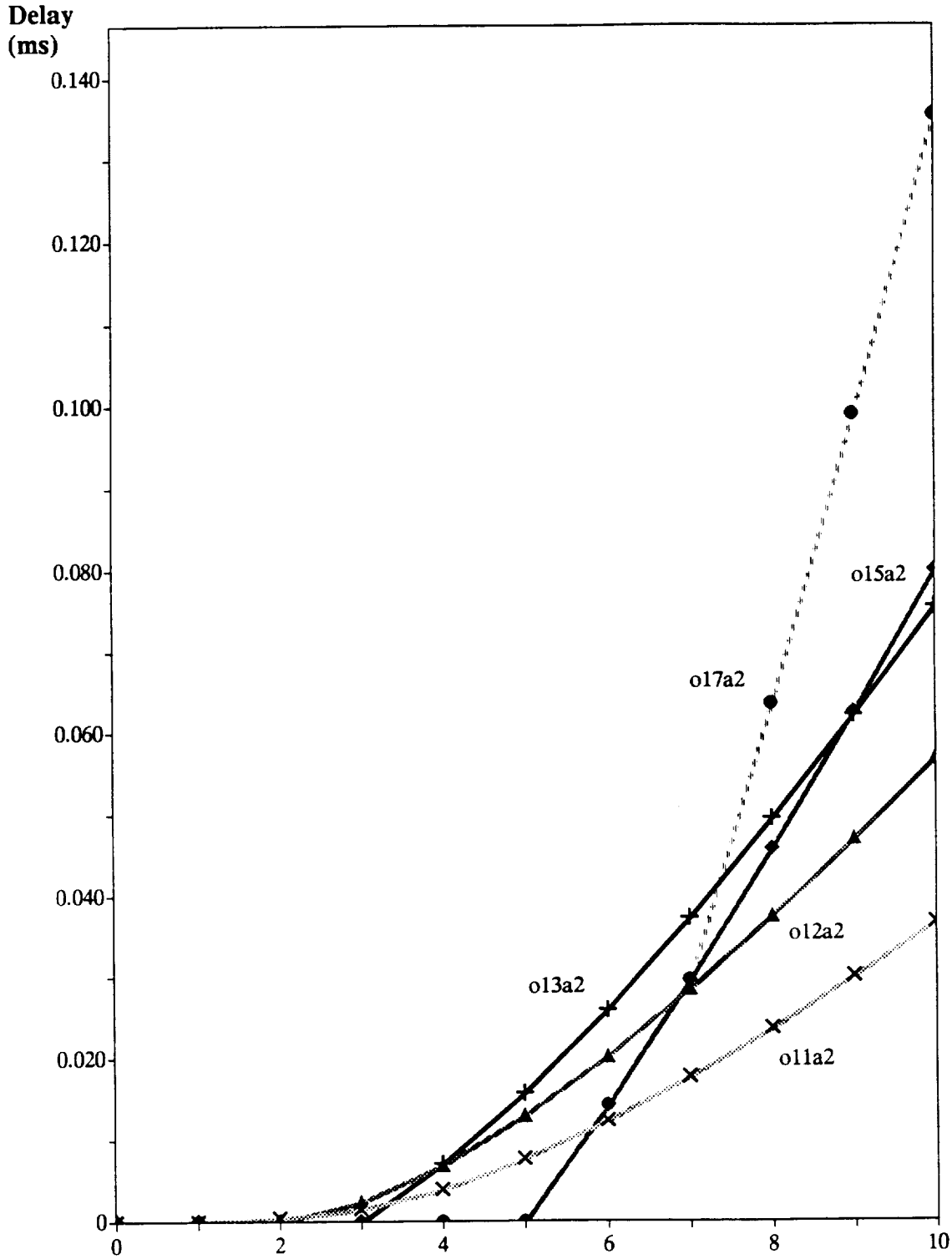


Figure 6.29 **Mean Delay in Milliseconds vs. Offered Load in Megabits for a Packet Length of 1 Slot Time for Scheduled Traffic on Ethernet**

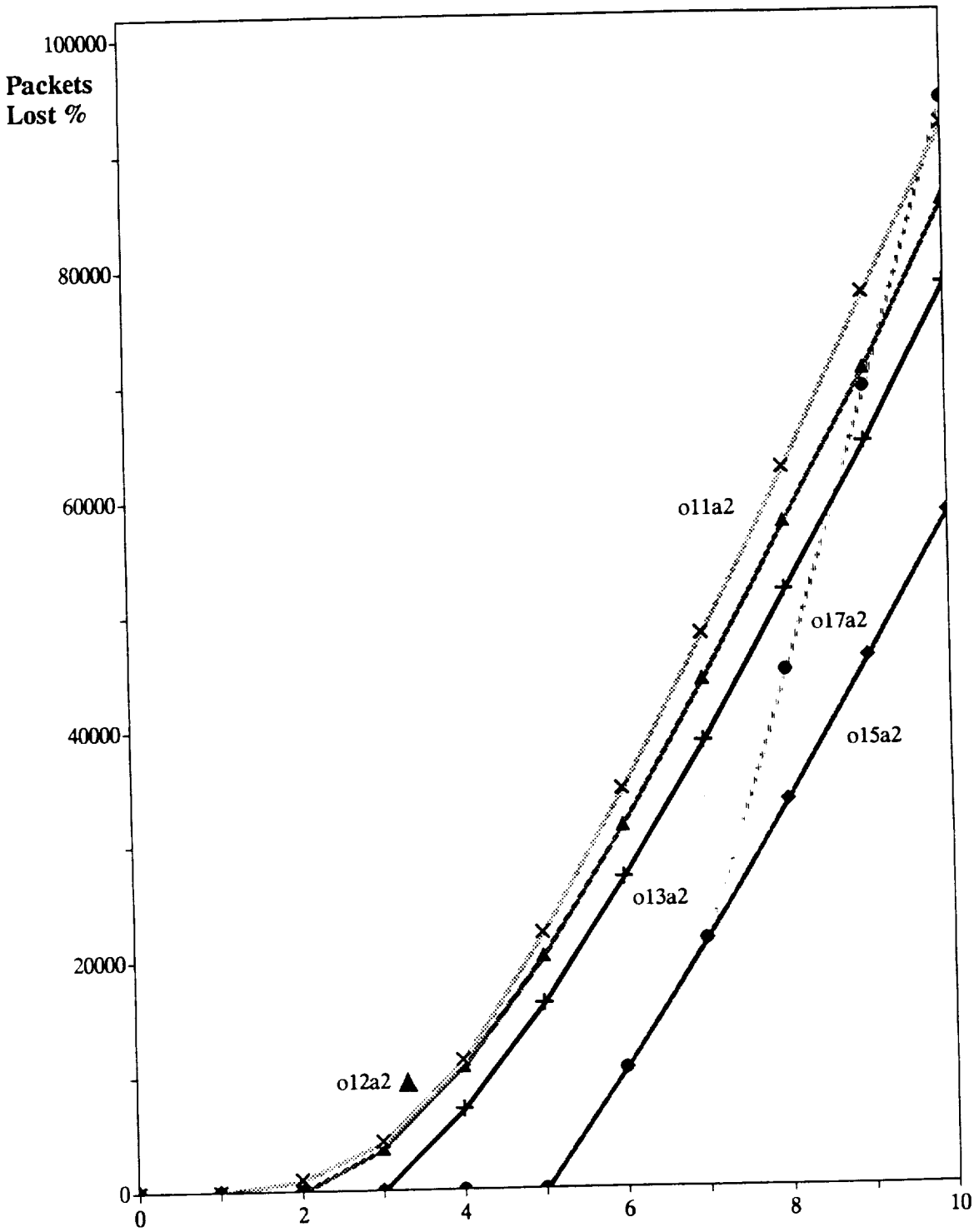


Figure 6.30 Packets Lost vs. Offered Load in Megabits for a Packet Length of 1 Slot Time for Scheduled Traffic on Ethernet

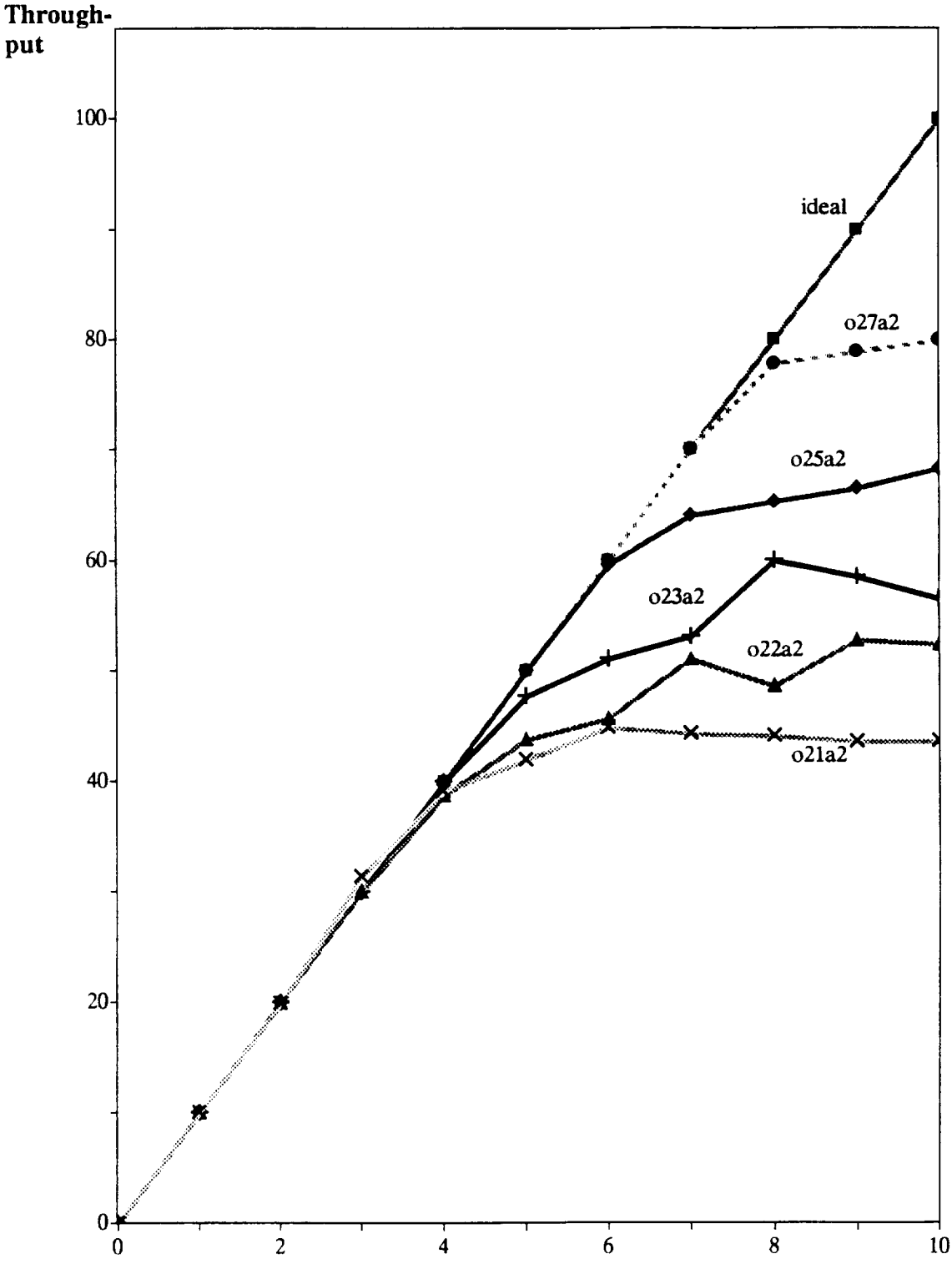


Figure 6.31 Throughput vs. Offered Load in Megabits for a Packet Length of 1 Slot Time for Contention Sync

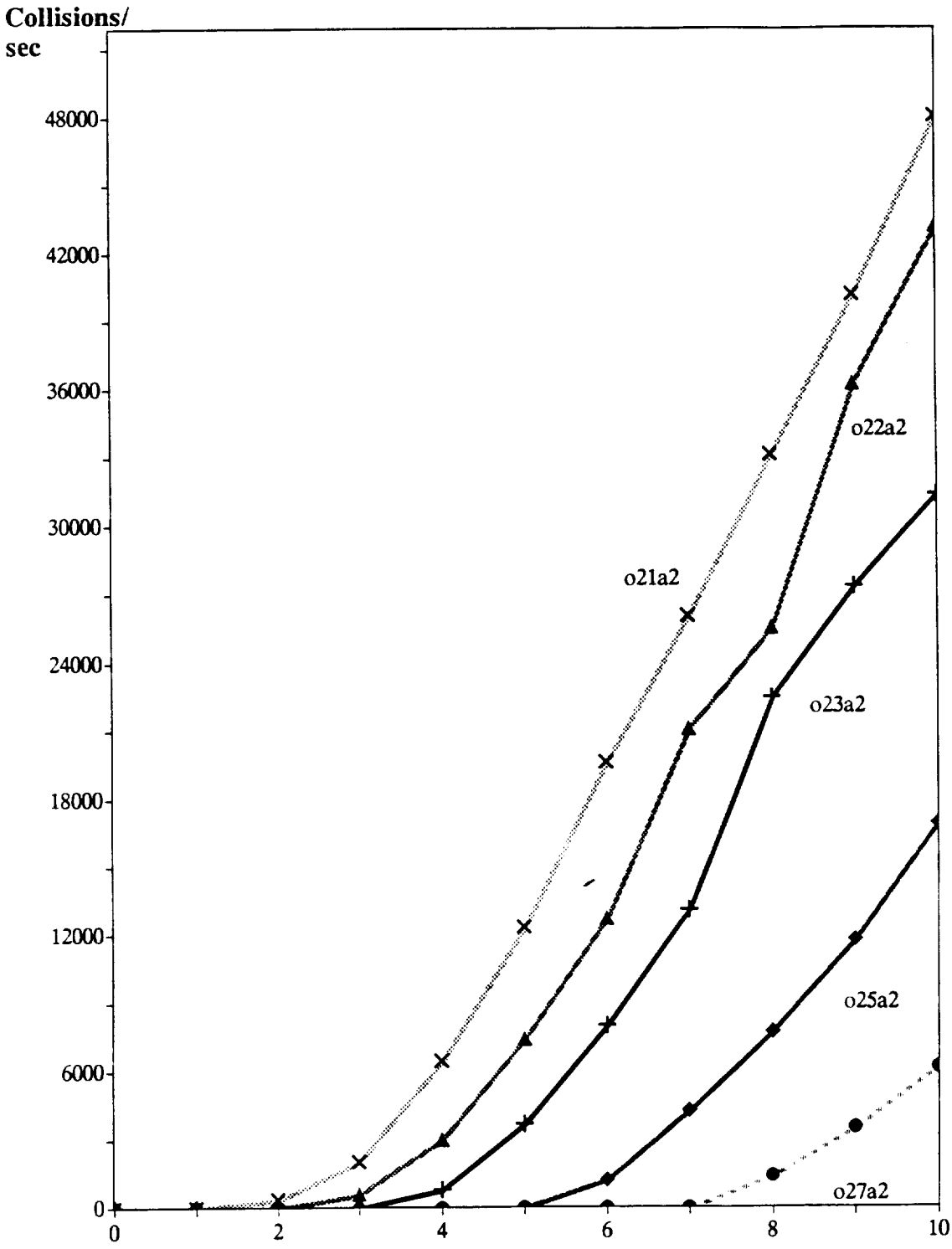


Figure 6.32 Collisions/second vs. Offered Load in Megabits for a Packet Length of 1 Slot Time for Contention Sync

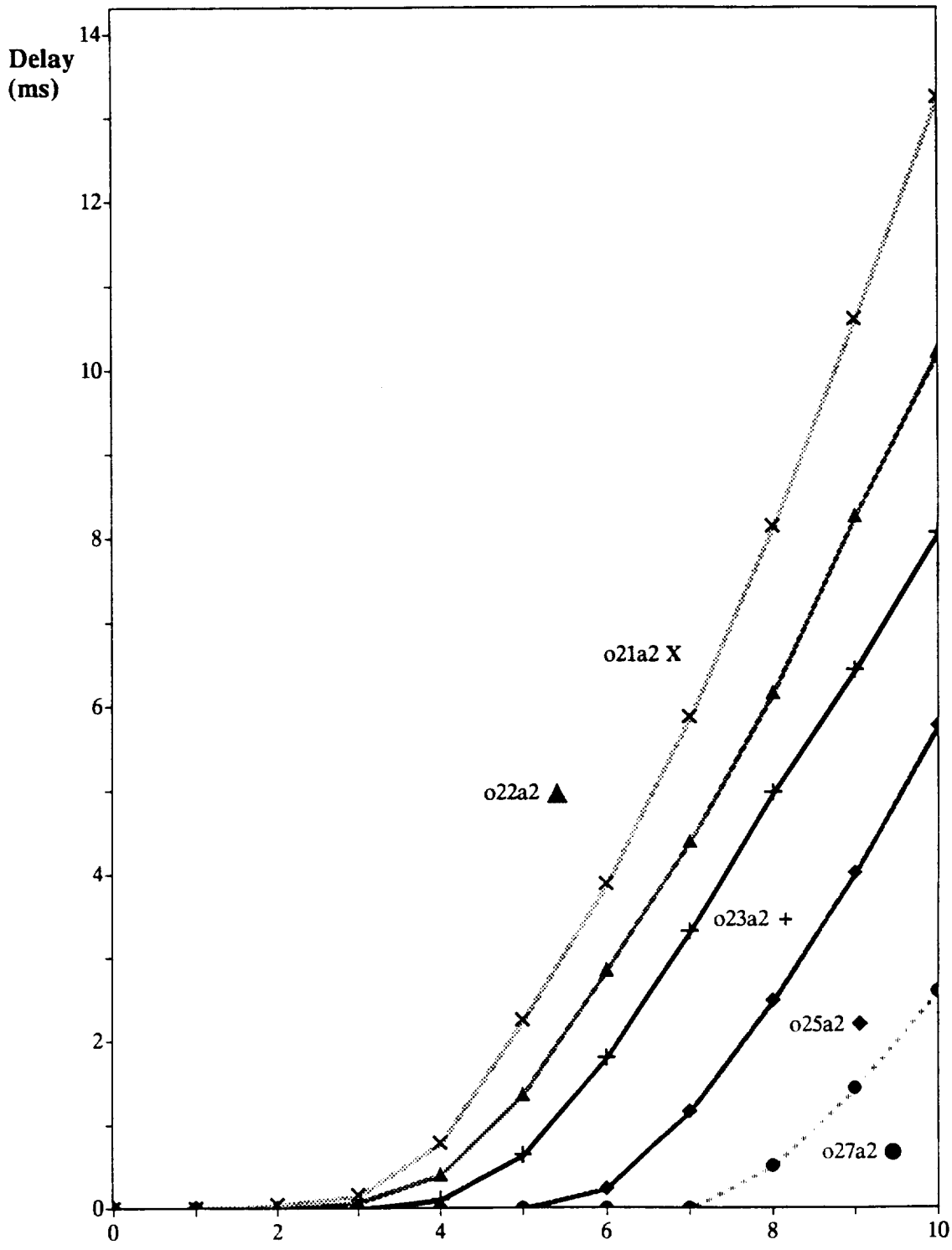


Figure 6.33 Delay vs. Offered Load in Megabits for a Packet Length of 1 Slot Time for Contention Sync

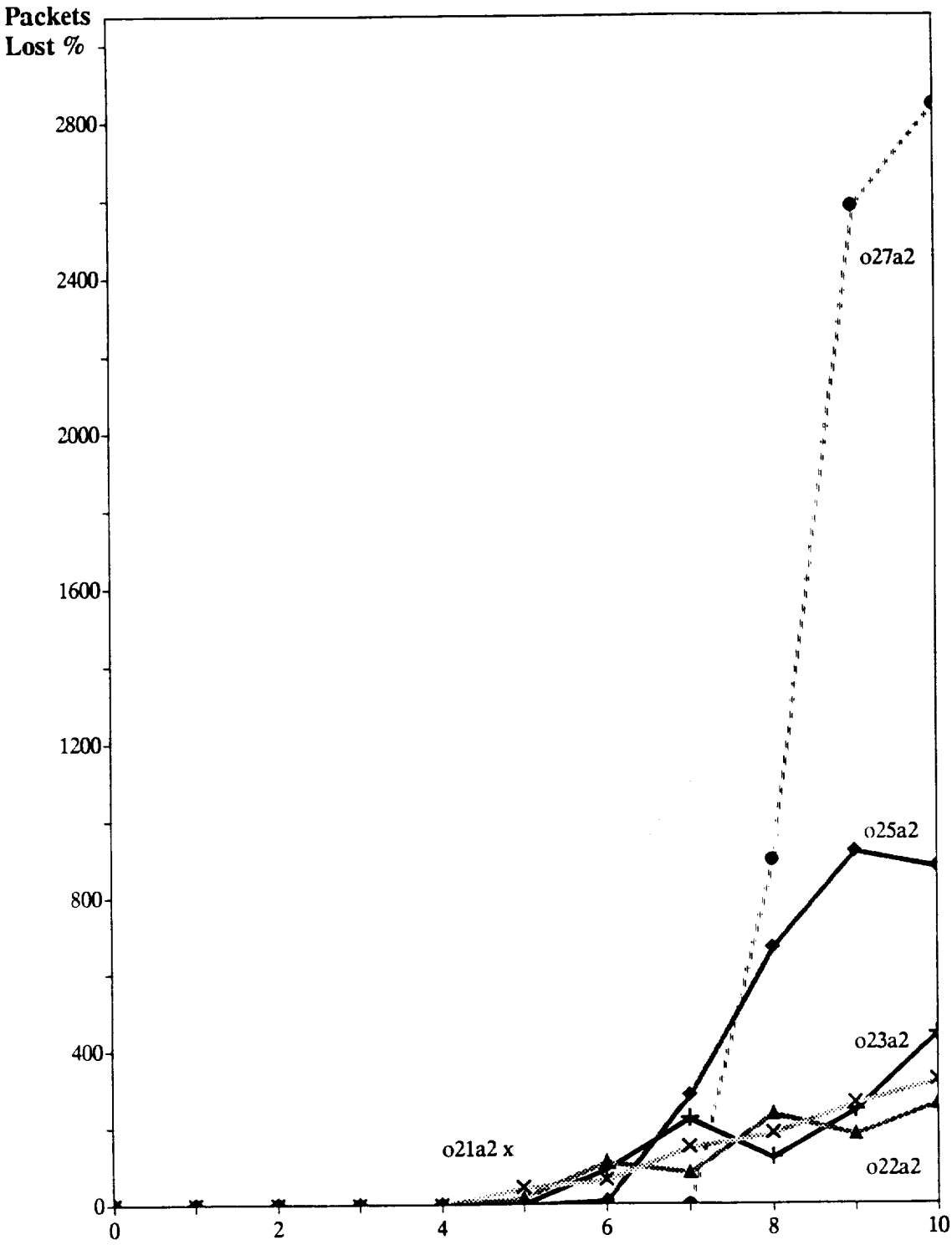


Figure 6.34 **Packets Lost due to Excessive Delay vs. Offered Load in Megabits for a Packet Length of 1 Slot Time with Contention Access for the Sync Packet**

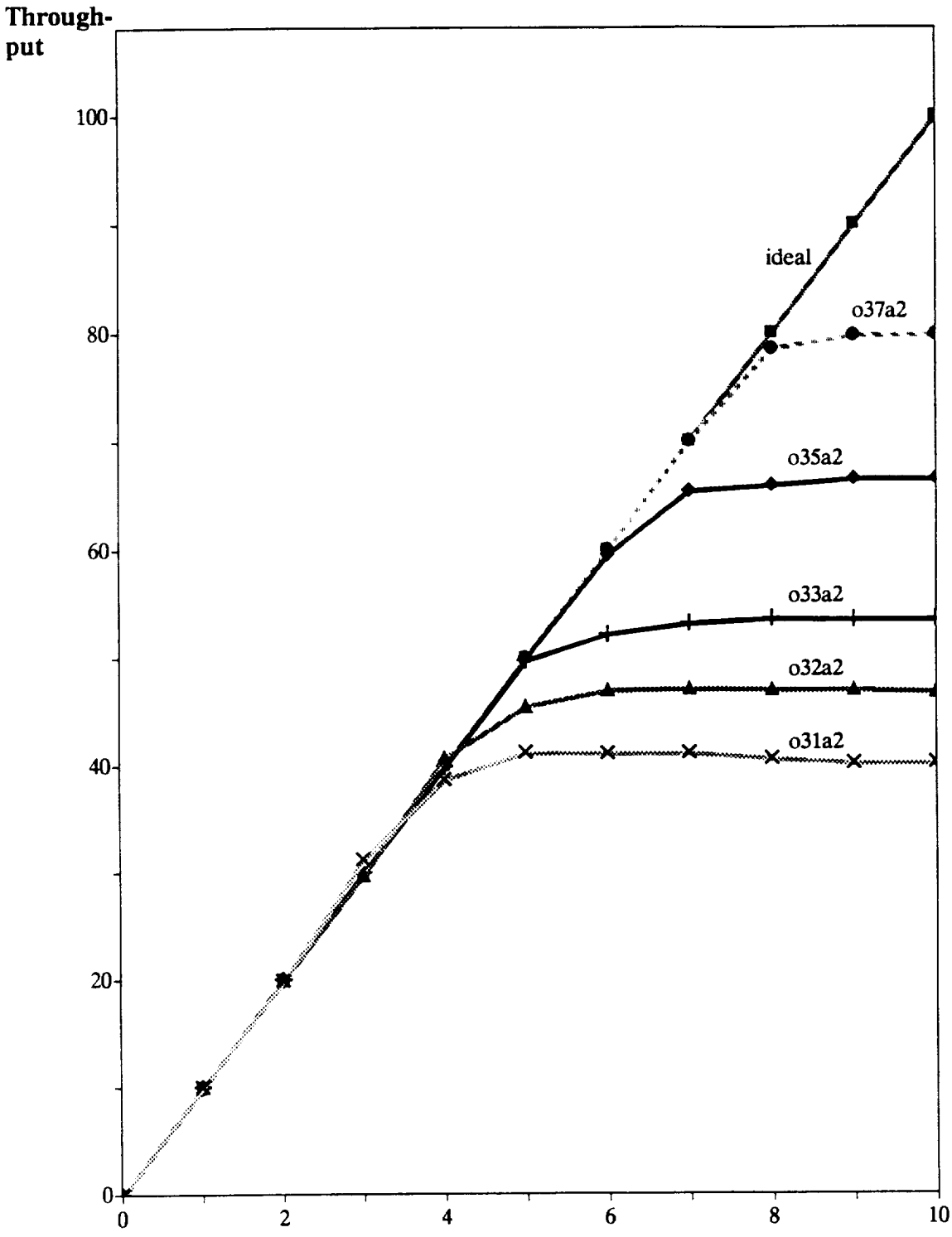


Figure 6.35 Throughput vs. Offered Load in Megabits for a Packet Length of 1 Slot Time (Non–Contention Sync)

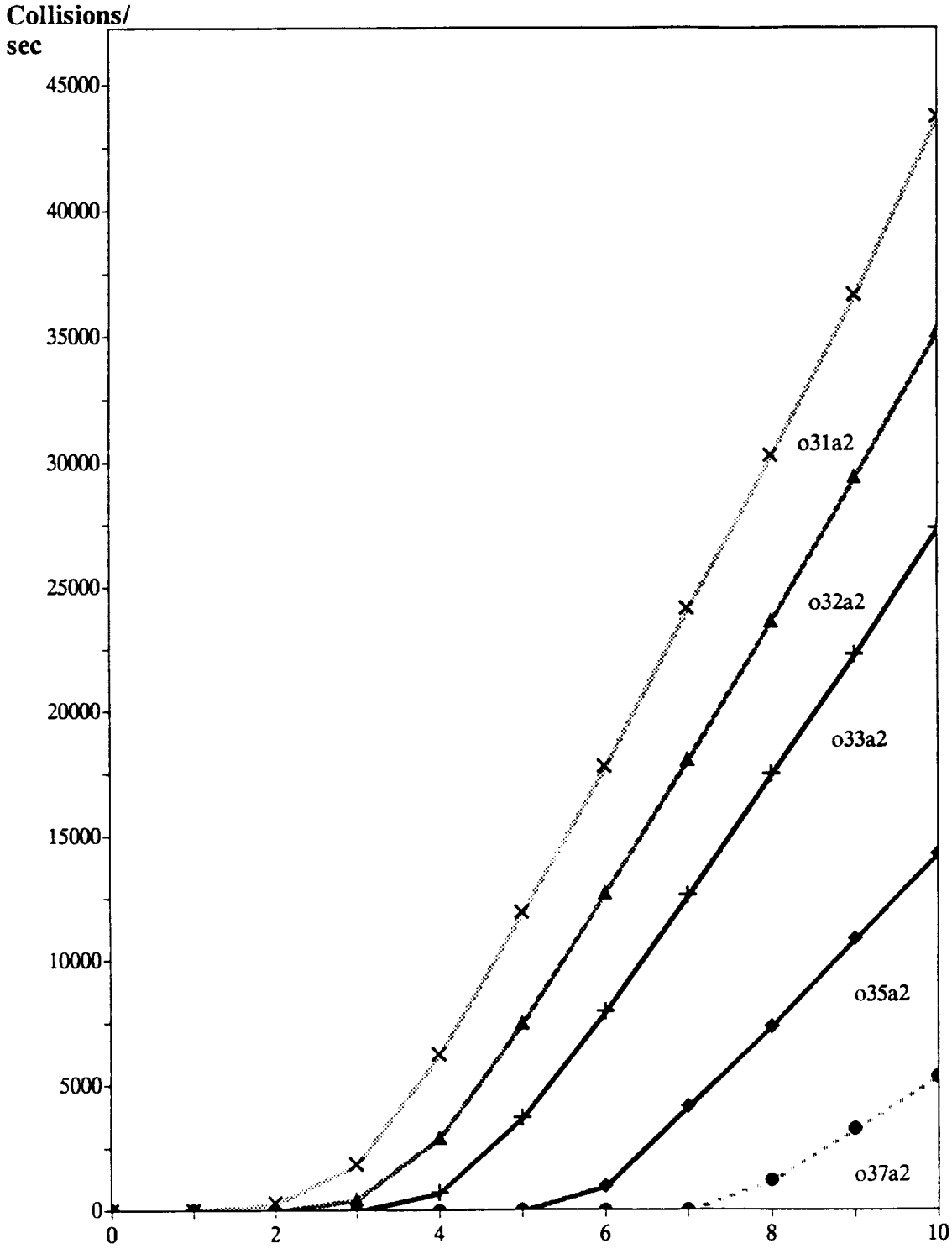


Figure 6.36 Collisions/second vs. Offered Load in Megabits for a Packet Length of 1 Slot Time (Non–Contention Sync)

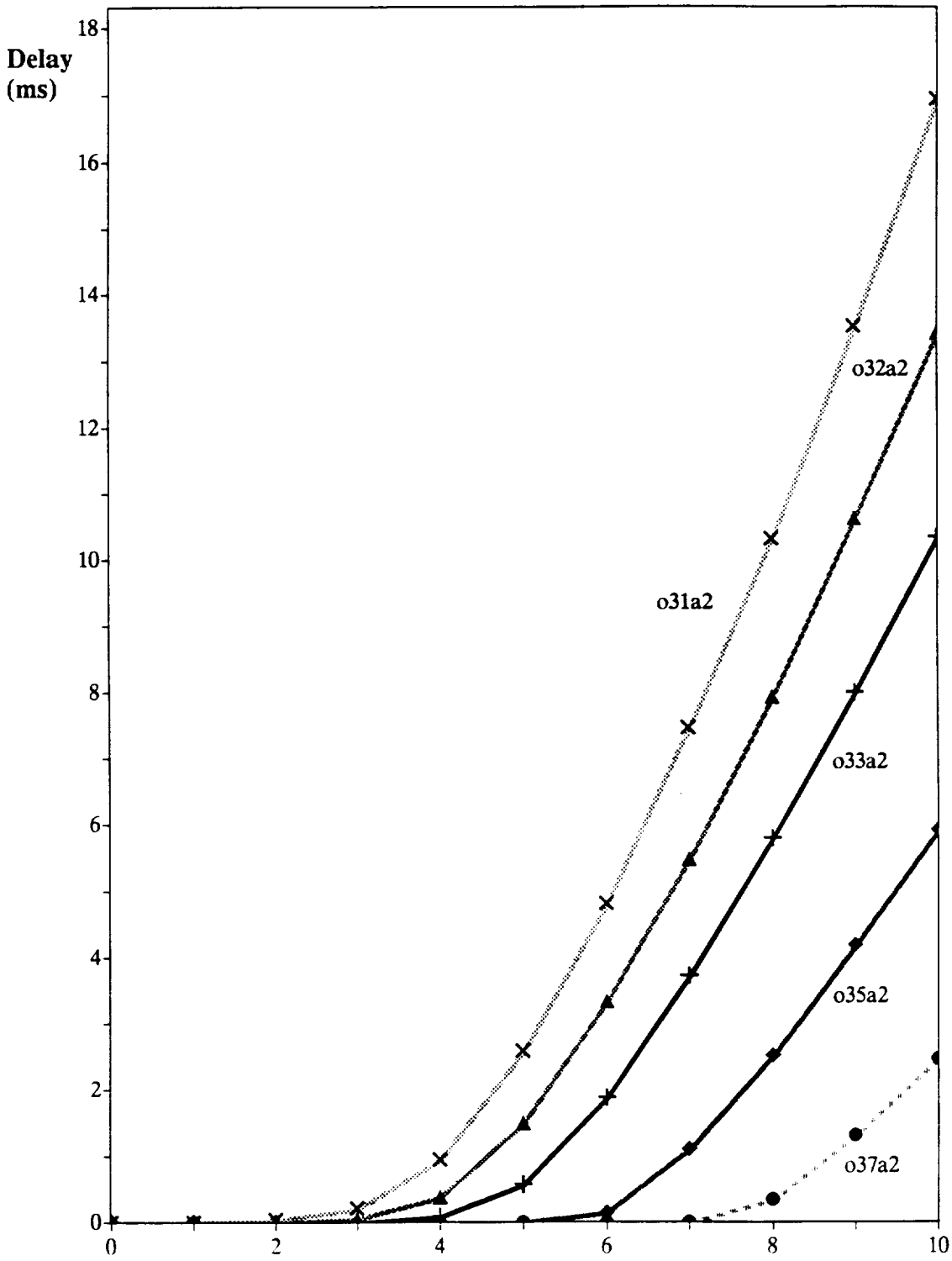


Figure 6.37 Delay vs. Offered Load in Megabits for a Packet Length of 1 Slot Time (Non-Contention Sync)

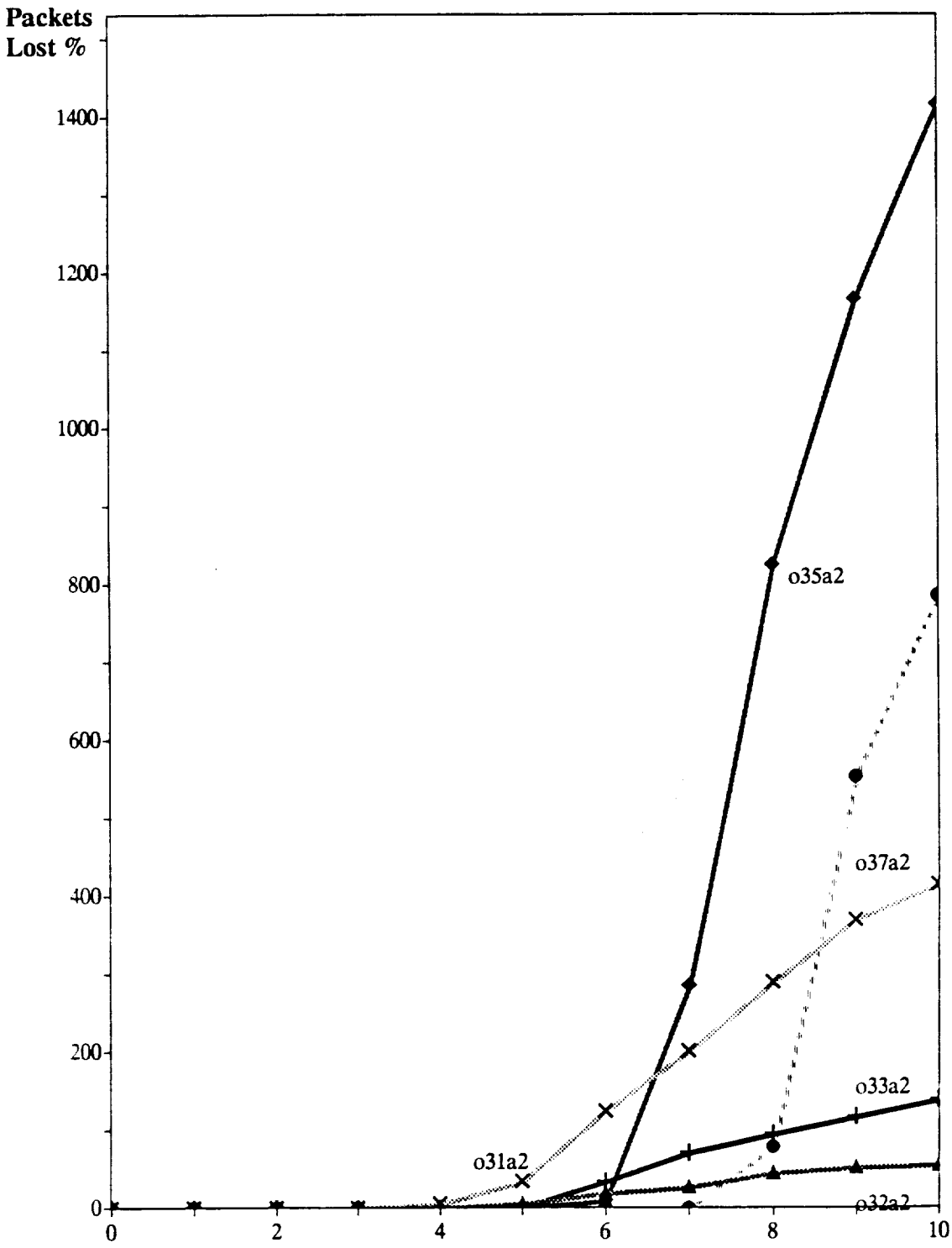


Figure 6.38 **Packets Lost vs. Offered Load in Megabits for a Packet Length of 1 Slot Time (Non-Contention Sync)**

7 CONCLUSIONS

The Modified Free Access protocol/hardware system has been developed to service communication systems that have a majority of Poisson distributed traffic with some periodic time critical packets. The protocol was implemented with readily available Ethernet hardware. This allows the protocol to be used economically in many systems. Several additional station types were introduced to be used with the Modified Free Access protocol. These include the pure Ethernet station, a 1-persistent CSMA/CD station, and the limited station.

There was no requirement to develop software for the system as commonly available host and terminal software can be used. Ethernet transport layer software can easily drive a MFA terminal since it will only see an Ethernet terminal. The modifications of the standard terminal hardware will only affect the terminal's transmitter section and this is hidden from the host.

7.1 PERFORMANCE OF THE MFA PROTOCOL

In order to study the performance of the MFA protocol a mathematical model and a computer simulation of the protocol were developed. The math model is limited in that only the present state of the system is known, namely the number of stations in backlog. The offered load, σ, and the probability that a backlogged station will sense the channel, ν, are the two input variables for the model. Throughput and average delay are the outputs of the model. The use of a fixed ν does not accurately represent the Ethernet and MFA backoff algorithms but is a fair approximation. If an appropriate method to determine $\nu(\sigma,M)$ could be found, it would be a valuable addition to the model.

There are four different simulator programs. The first is to verify the software simulation of the Ethernet protocol. The output was verified by comparison to Gonsalves' Ethernet measured data[GONS87]. The comparison between simulation and Gonsalves measured data is shown on Figures 7.1 and 7.2. The simulators' improved performance for small packet sizes is due to the difference in length of the networks. The second program is to show the performance of the Ethernet protocol when scheduled and Poisson traffic are generated by the stations. The third program is a simulation of the MFA protocol when the sync packet contends for the bus to begin the scheduled period. The fourth program is a simulation the MFA protocol when the sync packet has exclusive access to the bus as soon as the bus is clear after the beginning of a cycle.

The output of simulation one, the Ethernet simulation, compares well with Gonsalves' results as shown in Figures 7.1 and 7.2. The output values of simulation two indicated much poorer delay characteristics for the Ethernet protocol with a scheduled load as compared to the Ethernet protocol with a Poisson load. This performance degradation is as expected due to the increase in average collision rate for the scheduled loads.

The differences in outputs for simulations three and four are minor for all parameters except for average and maximum delay of scheduled traffic. These values are considerably lower in simulation four compared to simulation three's values. This indicates that the Td timer is valuable when the application requires minimal delay for scheduled data.

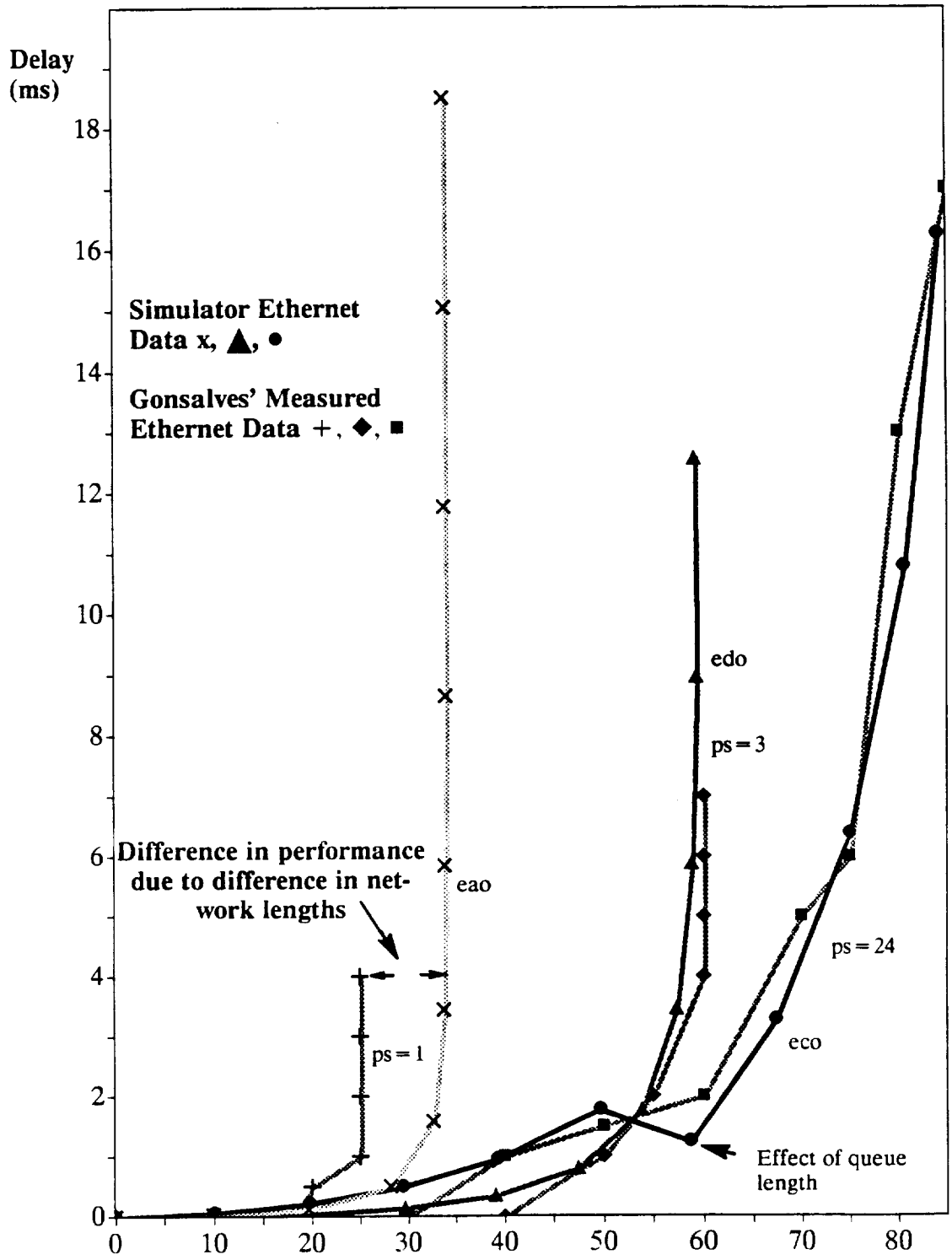


Figure 7.1 **Mean Delay in Milliseconds vs. Throughput for Various Packet Sizes. (1, 3 and 24 Slot Times)**

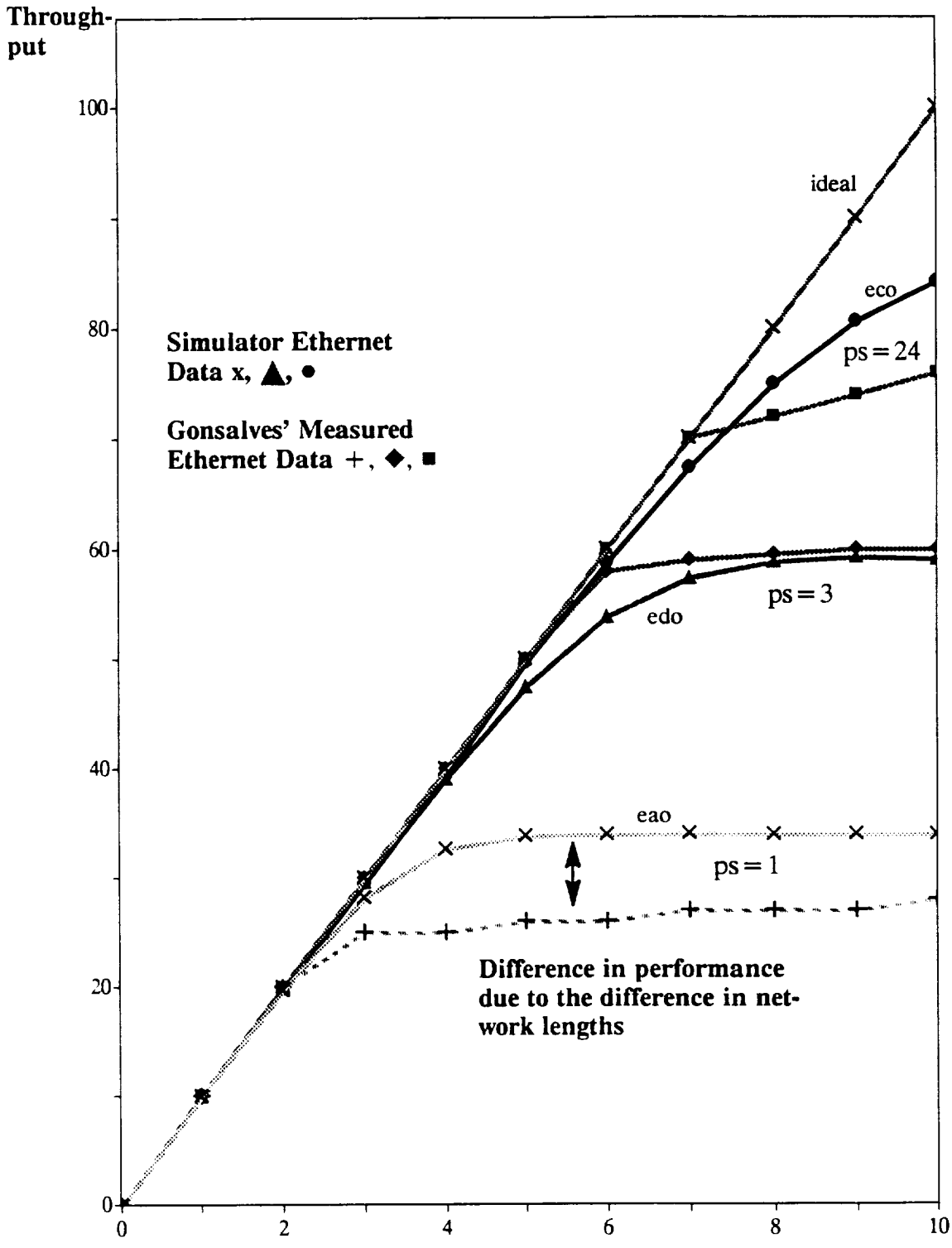


Figure 7.2 **Throughput vs. Offered Load in Megabits. (1, 3 and 24 Slot Times)**

A comparison of Ethernet with a Poisson load, Ethernet with one and three megabit scheduled mixed with a Poisson load and the MFA protocol with one and three megabit scheduled loads mixed with a Poisson load is shown in Figures 7.3 and 7.4. The MFA protocol performs as well as the Ethernet protocol for all offered loads. The Ethernet protocol does not perform well for the mixed scheduled and Poisson load.

The simulator's parameters were set to standard Ethernet values. The network was considered to be short so the end to end propagation time was considered negligible. The idle time after each transmission was considered as part of the transmission. The simulator and math model yielded good results and were in agreement with several previously published papers. The simulator was run with a slot time equal to the round trip propagation delay to speed execution. A finer grain simulation can be run with this simulator but this would take an excessive amount of time.

7.2 APPLICATIONS OF THE MFA PROTOCOL

The Modified Free Access (MFA) protocol is applicable to any system that has a mixture of scheduled data and Poisson. If a system has many stations that generate Poisson traffic and only one station that must have immediate access to the bus on a periodic basis then the Ethernet protocol may not be used due its CSMA/CD nature. The MFA protocol allows many of the benefits of Ethernet for the Poisson stations and still allows for the delivery of the time critical periodic traffic. Compared to Ethernet, the MFA protocol provides improved throughput versus offered load and throughput versus delay characteristics for any amount of scheduled load. There is a slight overhead for the sync packet but this can be kept very small. The MFA protocol should be chosen over Ethernet for any application that has periodic traffic and

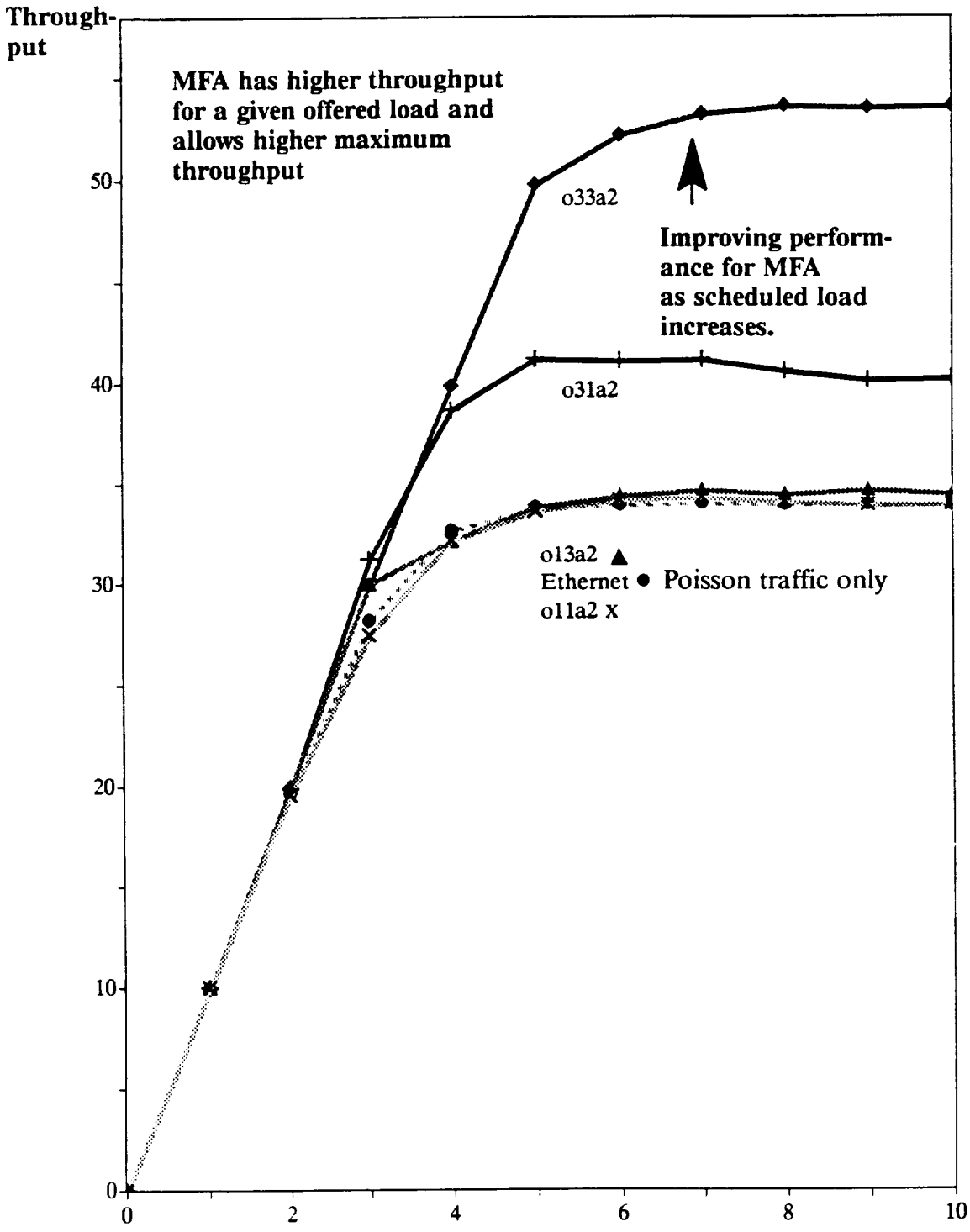


Figure 7.3 Throughput vs. Offered Load in Megabits for a Packet Length of 1 Slot Time for Ethernet and MFA Protocols

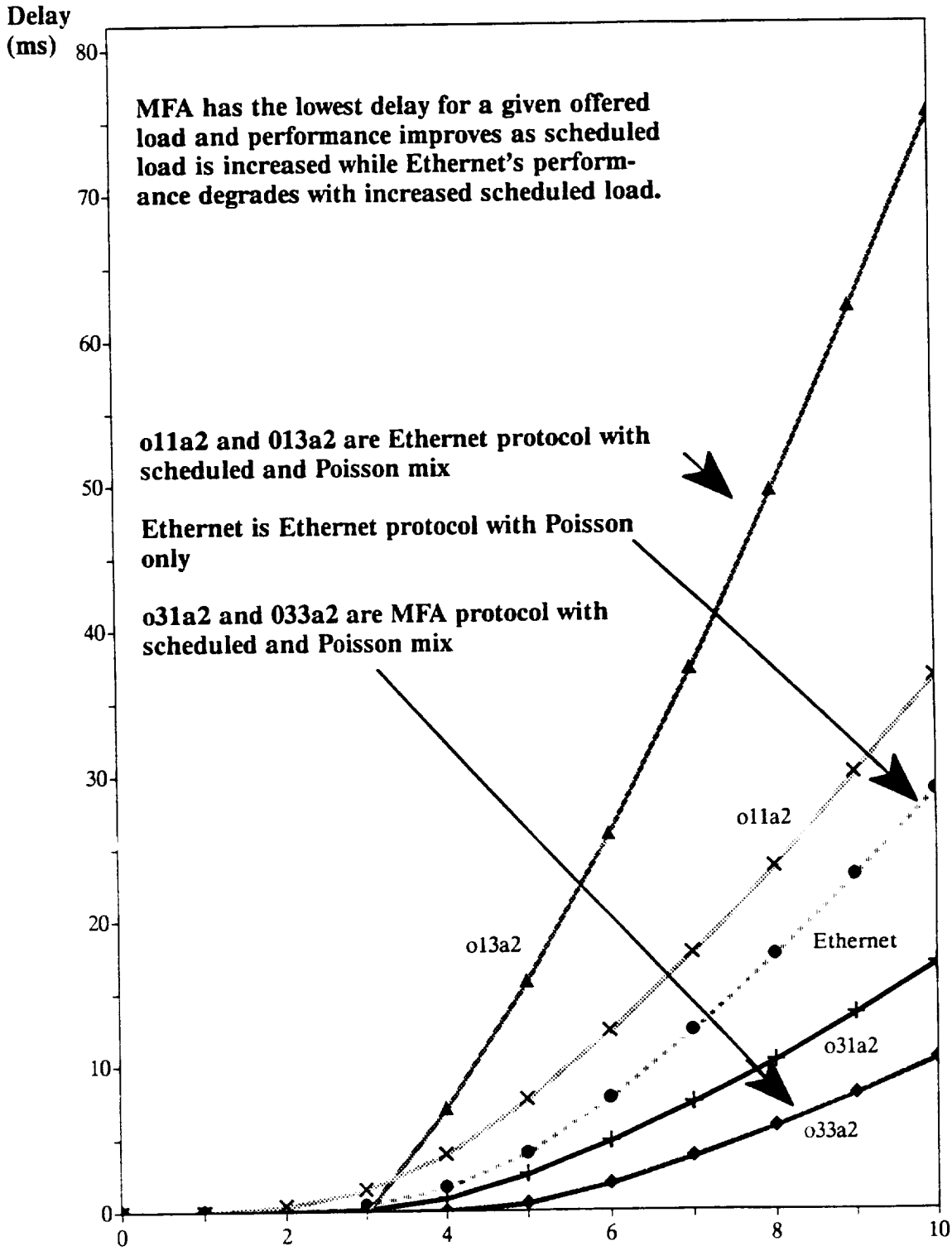


Figure 7.4 Mean Delay in Milliseconds vs. Offered Load in Megabits for a Packet Length of 1 Slot Time for Scheduled Traffic on Ethernet and MFA Protocols

Poisson traffic.

Table 7.1 is a comparison of the MFA, Ethernet, HYPERchannel, and ProNET-10 protocols. The MFA protocol resides in level 1, the physical layer, of the seven layer model. The MFA protocol uses the second level Ethernet specification and is also compatible with all Ethernet level three through seven software.

TABLE 7.1 COMPARISON OF PROTOCOLS

PROTOCOL:	MFA	Ethernet	MIL-STD-1553B	ProNET-10
BANDWIDTH	10	10	1	10
BOUNDED ACCESS TIME	Y	N	Y	Y
ARCH	BUS	BUS	BUS	RING
EXPANSION	EXCEL	EXCEL	GOOD	POOR
HARDWARE AVAILABLE	Y	Y	Y	Y
MAX # STATIONS	1024	1024	31	1024
ACCESS TYPE	HYBRID CSMA/CD	CSMA/CD	COMMAND/ RESPONSE	TOKEN
AVERAGE DELAY FOR POISSON SCHEDULED MIX	LOW	MEDIUM	HIGH	MEDIUM

7.3 RECOMMENDATIONS FOR FURTHER WORK

The development of a protocol that only reserves a short starting period for each station on a network is an obvious extension of the MFA protocol. This would be useful in some applications but would require a considerable amount of additional hardware. The extension of the MFA concept of scheduled and free access periods

could be applied to many other protocols. The application of similar ideas to FDDI and a new HYPERchannel system that operates at 100 megabits/second are in the development stage.

The development of an expression of $\nu(\sigma,M)$ would be beneficial to both the MFA and Ethernet math models. Additionally, further study of the Ethernet protocol's performance for non-Poisson offered loads should be made.

APPENDIX A SIMULATOR OUTPUT

A.1 COLUMN HEADINGS

The columns of Table A.1 will be labeled by the following system:

ρ = offered load
D = mean delay (milliseconds)
CR = collision rate (collisions/second)
B = average backoff period (slot–times)
T = total thoughput
T_i = information throughput
LQ = percentage of packets lost due to insufficient queue space
LD = percentage of packets lost due to excessive delay

Note: A subscript "p", such as X_p, indicates that the quantitiy X is only for packets with a poisson distributed arrival rate. Similarly, a suscript "s", such as X_s, indicates that the quantity X is only for packets scheduled to arrive at the beginning of a cycle.

A.2 SIMULATION PARAMETER DESIGNATORS: $N_1N_2LN_3$

For data labeled with a four character designator, $N_1N_2LN_3$:

N_1 indicates the way in which scheduled data appeared: 1 indicates each packet of scheduled data must contend for the channel using the Ethernet protocol; 2 indicates all scheduled data is contained in one long packet, which must contend for the network; and 3 is like 2, except that the long packet containing the scheduled data, without having to contend, gains access to the channel as soon as it is open at the beginning of a cycle.

$N_2 \times 10$ indicates the percentage of slots of the cycle that are used for scheduled data.

L indicates the packet length: "a" indicates a packet length of 1 slot-time (64 bytes); "b", 8 slot-times; and "c", 24 slot-times.

N_3 is the tens exponent of the length of the cycle, that is, the cycle length for the simulation is 10^{N_3} slot-times long.

For example, the simulation parameter designator 12a3 would indicate that each packet of the scheduled data must contend for the network ($N_1 = 1$), 20% of the cycle will be used for scheduled data ($N_2 = 2$), the packets are 1 slot-time long ($L = \text{a}$), and the cycles are $10^3 = 1000$ slot-times long ($N_3 = 3$).

TABLE A.1 Simulation Results

ρ	D	CR	B	T	LQ	LD
		Ethernet, packet-length = 1 slot-time				
0.1	0.03	188.04	2.68	9.96	0.36	0.00
0.2	0.12	1260.91	3.69	19.64	0.72	0.00
0.3	0.50	4335.04	6.82	28.19	2.02	0.00
0.4	1.58	9590.42	12.08	32.64	5.15	0.01
0.5	3.43	15857.17	16.42	33.80	8.19	0.06
0.6	5.84	22549.45	19.43	33.92	10.09	0.22
0.7	8.64	29794.13	21.12	34.00	10.91	0.34
0.8	11.76	37218.55	22.76	33.94	11.29	0.43
0.9	15.05	44940.77	23.73	33.97	11.24	0.57
1.0	18.50	52753.90	24.63	33.91	11.05	0.70
3.0	22.81	60823.32	29.65	33.42	29.98	1.61
5.0	27.18	70290.27	30.63	33.66	33.42	1.86
7.0	31.57	79964.12	31.26	33.94	31.63	1.93
10.0	35.93	89639.50	31.83	34.15	30.95	2.07
		Ethernet, packet-length = 3 slot-times				
0.1	0.02	22.22	2.31	10.01	0.26	0.00
0.2	0.06	162.09	2.83	19.84	0.29	0.00
0.3	0.15	578.77	3.58	29.61	0.42	0.00
0.4	0.35	1498.28	5.00	39.07	0.56	0.00
0.5	0.80	3182.36	7.47	47.48	0.98	0.00
0.6	1.77	5803.59	11.48	53.83	1.77	0.01
0.7	3.46	9184.17	15.75	57.35	2.92	0.05
0.8	5.89	12979.90	19.94	58.84	4.02	0.19
0.9	8.96	17041.48	23.21	59.27	4.97	0.40
1.0	12.56	21263.41	25.70	59.09	5.71	0.68
3.0	18.54	26291.42	37.50	57.43	24.28	3.66
5.0	24.77	31859.15	39.64	56.84	29.73	4.70
7.0	31.14	37827.19	40.37	56.88	29.33	5.06
10.0	37.52	43784.88	41.39	56.82	29.58	5.41

ρ	D	CR	B	T	LQ	LD
	Ethernet, packet-length=8 slot-times					
0.1	0.03	4.67	2.35	10.06	0.26	0.00
0.2	0.09	36.00	2.68	19.93	0.27	0.00
0.3	0.21	132.45	3.36	29.75	0.32	0.00
0.4	0.41	355.32	4.39	39.38	0.39	0.00
0.5	0.79	792.38	6.26	48.73	0.46	0.00
0.6	1.49	1536.18	9.15	57.52	0.69	0.00
0.7	2.73	2653.88	13.04	65.07	1.02	0.01
0.8	4.86	4175.39	18.02	70.84	1.54	0.07
0.9	7.96	5966.85	23.60	74.09	2.20	0.41
1.0	12.01	7946.00	28.44	75.58	2.88	0.97
3.0	19.72	10873.38	48.72	75.64	19.29	10.95
5.0	27.88	13929.44	51.45	75.52	25.57	13.65
7.0	36.19	17086.43	52.24	75.60	26.44	14.95
10.0	44.53	20265.11	53.17	75.56	27.51	16.06
	Ethernet, packet-length=24 slot-times					
0.1	0.07	0.87	1.99	10.04	0.22	0.00
0.2	0.23	8.71	2.57	19.76	0.27	0.00
0.3	0.51	35.60	3.10	29.48	0.33	0.00
0.4	0.96	98.64	4.77	39.28	0.36	0.00
0.5	1.78	240.28	7.74	49.67	0.43	0.00
0.6	1.25	193.60	9.57	58.74	2.97	0.00
0.7	3.28	516.05	14.87	67.36	2.26	0.00
0.8	6.39	994.21	20.57	74.95	2.20	0.12
0.9	10.79	1610.15	27.89	80.63	2.74	0.84
1.0	16.26	2357.13	32.97	84.19	3.09	2.02
3.0	25.06	3692.56	59.94	89.40	23.19	30.44

ρ	D	CR	B_p	B_s	T	LQ_p	LQ_s	LD
11a2								
0.2	0.40	1565.67	5.51	6.65	19.57	3.88	0.02	0.00
0.3	1.54	5647.87	8.66	10.07	27.47	6.88	1.00	0.00
0.4	3.91	11795.36	12.86	14.58	32.15	10.38	4.93	0.03
0.5	7.66	18591.11	17.09	18.58	33.61	13.25	12.42	0.12
0.6	12.31	25830.13	20.13	20.55	34.14	14.43	17.79	0.27
0.7	17.68	33324.82	22.12	22.83	34.32	14.69	22.68	0.41
0.8	23.64	40794.31	23.84	23.22	34.10	14.55	26.36	0.58
0.9	30.04	48474.99	24.97	24.18	33.92	14.18	29.06	0.69
1.0	36.67	56601.30	25.75	24.66	33.93	13.56	30.10	0.80
11a3								
0.2	4.71	2208.72	15.07	23.10	17.87	21.13	0.00	0.00
0.3	8.97	6127.38	11.92	24.54	24.85	14.91	0.01	0.00
0.4	13.33	11809.51	12.59	25.61	30.52	13.15	0.10	0.01
0.5	18.56	18620.52	16.43	26.24	33.34	13.91	0.47	0.09
0.6	24.86	25768.82	20.16	26.67	34.15	14.94	1.10	0.23
0.7	31.93	33190.96	22.44	26.66	34.33	15.18	1.47	0.42
0.8	39.66	40781.78	24.06	27.05	34.23	14.99	1.89	0.59
0.9	48.00	48623.52	25.67	26.71	34.15	14.55	2.38	0.71
1.0	56.51	56968.67	26.48	26.99	34.33	13.77	2.42	0.85
11a4								
0.2	6.47	1219.90	10.99	30.12	13.84	11.12	51.05	0.10
0.3	10.82	3851.37	8.40	29.68	22.21	7.03	52.42	0.08
0.4	14.42	8450.16	8.73	30.02	29.47	6.50	53.38	0.10
0.5	18.43	14670.50	13.51	29.43	33.13	8.93	54.11	0.12
0.6	23.07	21420.38	17.48	28.78	34.03	11.19	55.19	0.27
0.7	28.26	28569.97	20.11	28.93	34.35	12.32	55.63	0.38
0.8	33.84	35830.44	22.08	29.27	34.22	12.79	56.21	0.52
0.9	39.73	43573.18	23.50	29.24	34.30	12.72	56.61	0.59
1.0	45.88	51022.57	24.62	30.11	33.97	12.57	56.66	0.72
11b2								
0.2	0.07	51.94	2.79	2.62	25.92	0.83	0.00	0.00
0.3	0.22	216.74	3.42	3.16	35.83	0.70	0.00	0.00
0.4	0.50	562.34	4.80	4.25	45.43	0.78	0.01	0.00
0.5	1.03	1196.94	6.93	5.64	54.64	0.97	0.02	0.00
0.6	2.13	2203.47	10.83	8.57	63.10	1.29	0.81	0.01
0.7	4.19	3641.60	15.75	11.99	69.83	1.97	2.51	0.06
0.8	7.72	5366.32	21.12	17.71	73.73	2.78	9.92	0.28
0.9	12.83	7259.53	27.26	22.39	75.26	3.76	19.22	0.77
1.0	19.17	9321.35	31.12	27.55	76.05	4.41	29.29	1.50

ρ	D	CR	B_p	B_s	T	LQ_p	LQ_s	LD
11b3								
0.2	0.41	122.12	9.23	6.56	20.34	0.94	0.00	0.00
0.3	1.06	408.08	9.64	8.45	30.10	0.95	0.00	0.00
0.4	1.97	895.29	9.74	11.06	39.68	0.98	0.00	0.00
0.5	3.21	1628.41	10.79	14.06	48.77	1.18	0.00	0.00
0.6	4.89	2674.23	12.53	16.37	57.35	1.41	0.00	0.00
0.7	7.22	4064.73	15.45	20.22	64.84	1.84	0.08	0.02
0.8	10.68	5785.07	20.02	24.45	70.32	2.51	0.20	0.14
0.9	15.49	7764.78	25.16	28.97	73.83	3.28	0.87	0.52
1.0	21.43	9851.03	30.04	32.35	75.39	4.05	1.68	1.30
11c4								
0.2	3.02	54.07	23.76	10.11	18.36	0.80	16.35	0.00
0.3	5.59	146.93	22.31	12.94	27.65	0.71	22.43	0.00
0.4	8.13	284.63	19.90	16.00	36.92	0.72	27.37	0.00
0.5	10.79	481.91	19.15	18.50	46.07	0.75	32.01	0.02
0.6	13.79	762.02	18.86	21.02	55.04	0.81	35.57	0.07
0.7	17.32	1134.90	20.23	24.42	63.18	0.92	40.50	0.18
0.8	21.54	1636.58	23.02	27.04	71.01	1.08	44.84	0.41
0.9	26.74	2269.64	26.12	32.40	77.67	1.33	49.83	0.78
1.0	33.04	3015.52	31.19	37.67	82.60	1.67	53.02	1.64
12a2								
0.3	2.41	4254.72	10.68	12.51	27.59	20.17	1.79	0.00
0.4	6.87	10667.88	14.47	16.94	31.57	21.65	7.43	0.05
0.5	12.94	17921.34	18.55	20.23	33.40	21.46	14.07	0.19
0.6	20.20	25531.43	21.69	22.24	34.18	20.64	19.50	0.37
0.7	28.47	33235.06	23.27	24.10	34.29	19.62	24.34	0.55
0.8	37.43	41037.27	25.40	24.76	34.22	18.55	27.27	0.72
0.9	46.94	49117.78	26.06	26.12	34.29	17.19	30.66	0.84
1.0	56.80	57136.02	26.98	26.29	34.16	16.17	32.32	1.02
12a3								
0.3	9.38	3605.62	18.61	26.65	26.22	36.12	0.83	0.00
0.4	18.67	8874.31	15.84	27.34	31.14	26.10	2.44	0.04
0.5	28.28	15527.73	17.72	27.19	34.20	22.44	5.38	0.11
0.6	38.85	22649.03	20.52	27.76	35.11	21.22	10.06	0.31
0.7	50.36	29917.99	23.60	27.58	34.89	20.45	13.89	0.56
0.8	62.56	37712.06	25.33	28.27	34.99	19.06	16.79	0.69
0.9	75.28	45444.18	26.95	27.86	34.74	17.90	18.68	0.88
1.0	88.41	53578.26	27.71	28.47	34.73	16.69	19.89	0.97

ρ	D	CR	B_p	B_s	T	LQ_p	LQ_s	LD
12a4								
0.3	10.25	2103.05	15.06	30.65	16.42	20.18	57.95	0.54
0.4	17.42	5758.10	9.64	30.34	23.79	12.18	58.73	0.38
0.5	23.60	11391.49	9.93	30.62	30.37	10.48	59.45	0.30
0.6	29.77	18087.28	13.67	30.76	33.42	11.66	60.59	0.34
0.7	36.58	25180.68	17.83	30.60	34.37	13.27	61.48	0.38
0.8	43.87	32461.16	20.89	29.86	34.45	14.04	61.73	0.59
0.9	51.59	39867.02	22.50	30.85	34.27	14.12	62.42	0.65
1.0	59.53	47767.96	23.69	31.22	34.37	13.75	62.58	0.78
12b3								
0.3	1.20	282.02	13.74	9.43	29.85	2.00	0.01	0.00
0.4	3.10	831.65	14.11	12.07	39.39	1.97	0.04	0.00
0.5	5.57	1670.54	14.86	14.23	48.53	2.09	0.08	0.00
0.6	8.68	2812.13	16.03	16.94	56.97	2.32	0.17	0.00
0.7	12.59	4316.65	17.24	19.94	64.80	2.52	0.92	0.01
0.8	17.78	6101.38	21.54	23.09	70.21	3.31	3.26	0.15
0.9	24.38	8086.51	25.64	26.70	73.67	4.04	7.26	0.52
1.0	32.46	10193.12	30.85	30.45	75.25	4.82	13.08	1.28
13a2								
0.4	7.07	6422.26	18.69	19.75	32.12	45.80	10.55	0.12
0.5	15.77	13874.26	21.07	22.46	33.83	36.22	16.24	0.36
0.6	25.96	21600.12	24.21	24.25	34.39	30.68	21.86	0.54
0.7	37.27	29541.85	25.51	25.92	34.64	26.70	26.32	0.66
0.8	49.50	37441.77	27.26	26.55	34.46	23.90	29.78	0.85
0.9	62.18	45886.29	27.67	27.18	34.68	21.37	31.63	0.99
1.0	75.53	53894.05	28.65	27.68	34.45	19.44	34.29	1.11
13a4								
0.4	12.80	2990.35	18.31	30.95	19.04	29.17	60.15	0.77
0.5	22.62	7491.81	11.87	31.02	25.43	18.20	61.25	0.66
0.6	30.99	13781.69	11.42	31.07	31.02	14.50	61.76	0.52
0.7	39.40	20998.13	14.94	30.93	33.72	14.73	62.58	0.50
0.8	48.37	28422.51	18.32	30.95	34.50	15.50	63.61	0.64
0.9	57.77	35954.30	20.81	31.36	34.63	15.71	63.97	0.72
1.0	67.51	43551.32	23.01	30.85	34.47	15.35	64.38	0.87
13b2								
0.4	0.16	187.23	4.00	3.54	41.90	1.62	0.00	0.00
0.5	0.53	649.01	5.77	4.45	51.46	1.54	0.01	0.00
0.6	1.40	1503.58	8.97	6.97	60.71	1.77	0.35	0.00
0.7	3.41	2784.62	13.98	11.19	68.13	2.58	2.17	0.03
0.8	7.29	4434.01	20.45	15.97	72.83	3.58	7.71	0.17
0.9	13.10	6269.38	25.59	21.01	74.83	4.64	16.55	0.58
1.0	20.64	8259.57	29.94	25.26	75.84	5.41	25.13	1.11

ρ	D	CR	B_p	B_s	T	LQ_p	LQ_s	LD
13c2								
0.4	0.23	45.69	3.68	2.74	57.90	1.83	0.00	0.00
0.5	0.83	181.58	7.77	3.59	67.45	1.62	0.01	0.00
0.6	2.44	485.37	14.85	6.54	76.62	1.78	0.41	0.03
0.7	6.15	998.86	24.40	10.39	83.43	2.47	3.09	0.28
0.8	13.06	1660.09	31.82	16.68	86.19	3.18	11.96	1.03
0.9	22.57	2384.90	35.57	22.59	87.10	3.52	23.37	1.89
1.0	33.89	3164.96	39.60	28.07	87.44	3.87	33.84	3.37
13c4								
0.4	5.58	139.80	29.32	12.91	31.82	1.73	26.95	0.07
0.5	10.88	333.90	32.99	16.05	39.44	1.66	33.41	0.33
0.6	16.12	593.12	34.08	18.89	47.20	1.59	38.80	0.63
0.7	21.49	927.92	34.31	20.67	54.65	1.56	43.76	1.02
0.8	27.18	1349.16	33.15	22.62	61.67	1.68	48.61	1.33
0.9	33.31	1863.02	31.84	25.19	68.08	1.72	54.45	1.58
1.0	40.03	2491.90	31.86	28.69	74.43	1.82	60.18	1.67
15a2								
0.6	14.25	7888.23	28.35	27.68	35.39	74.07	33.58	0.91
0.7	29.59	16041.03	27.92	28.45	35.39	51.31	37.37	1.05
0.8	45.71	24273.34	28.43	29.23	35.34	39.97	40.19	1.18
0.9	62.55	32675.81	29.78	29.42	35.31	32.90	42.29	1.29
1.0	79.97	41167.32	30.24	29.67	35.29	27.97	44.08	1.33
15a4								
0.6	16.34	4747.80	23.49	31.14	24.36	46.64	62.09	1.14
0.7	30.47	10746.31	17.28	31.30	28.72	30.44	63.01	0.94
0.8	43.33	18111.16	14.79	31.36	32.48	23.57	63.67	0.88
0.9	56.15	26075.72	16.67	31.67	34.32	21.32	64.54	0.89
1.0	69.23	33840.46	20.05	31.35	34.64	20.27	65.08	0.97
15b2								
0.6	0.58	539.19	9.94	6.19	65.49	3.39	0.33	0.00
0.7	2.98	1672.93	16.13	11.22	71.81	4.74	4.63	0.02
0.8	7.59	3149.09	21.35	15.66	74.28	5.79	14.28	0.15
0.9	14.11	4841.42	24.82	19.85	75.36	6.27	24.37	0.45
1.0	22.34	6703.12	28.75	23.50	75.94	6.82	33.05	0.96
15b4								
0.6	5.43	743.86	24.38	15.91	44.87	5.64	29.16	0.15
0.7	11.48	1669.56	24.70	18.86	49.86	4.63	37.16	0.35
0.8	18.06	2794.12	25.76	21.25	54.77	4.46	44.09	0.51
0.9	24.97	4130.47	26.38	24.07	59.80	4.32	49.95	0.69
1.0	32.23	5682.74	28.23	26.33	64.36	4.33	55.00	0.96

ρ	D	CR	B_D	B_S	T	LQ_D	LQ_S	LD
15c2								
0.6	0.77	164.85	13.78	3.67	81.78	3.04	0.05	0.01
0.7	4.36	654.75	30.31	7.95	88.22	3.98	2.95	0.38
0.8	11.69	1268.22	34.59	13.96	88.01	4.51	14.62	0.90
0.9	20.73	1935.19	36.61	18.06	88.09	4.56	24.84	1.52
1.0	31.32	2657.17	39.98	21.55	88.02	4.53	34.71	2.53
15c4								
0.6	6.68	225.95	33.12	13.85	45.25	2.56	29.28	0.14
0.7	13.41	509.22	34.13	17.07	51.42	2.52	35.63	0.44
0.8	20.38	872.07	39.21	19.66	57.33	2.57	41.15	1.25
0.9	27.62	1317.16	40.79	21.32	63.06	2.54	45.99	2.01
1.0	35.42	1842.14	39.48	23.48	68.20	2.58	51.94	2.37
17a2								
0.8	17.78	8360.64	30.33	29.90	35.88	81.90	50.52	1.24
0.9	36.41	16735.13	28.69	30.47	35.37	56.45	52.97	1.41
1.0	55.41	25360.10	30.58	30.52	35.45	42.96	54.35	1.52
17b2								
0.8	2.98	1061.89	18.88	11.99	74.28	6.78	9.62	0.03
0.9	8.01	2473.42	22.09	15.41	75.64	7.29	19.39	0.15
1.0	14.96	4102.16	25.68	18.85	75.99	7.63	29.34	0.41
17b4								
0.8	5.88	1046.03	25.35	15.97	58.39	8.07	29.74	0.22
0.9	12.82	2278.54	26.16	18.49	61.40	6.95	37.49	0.36
1.0	20.66	3679.97	27.45	21.65	64.33	6.23	44.98	0.61
17c2								
0.8	0.77	164.85	13.78	3.67	81.78	3.04	0.05	0.01
0.9	4.36	654.75	30.31	7.95	88.22	3.98	2.95	0.38
1.0	11.69	1268.22	34.59	13.96	88.01	4.51	14.62	0.90

ρ	D	CR	B_p	B_s	T	T_j	D_{ms}	D_{as}	LQ_p	LQ_s	LD_s	LD
25a4												
0.6	1.72	174.75	0.64	0.10	54.92	54.91	0.72	0.08	7.90	0.00	0.00	0.00
0.7	4.71	861.24	0.44	0.20	59.09	59.08	2.15	0.18	14.00	0.00	0.00	0.00
0.8	7.85	2502.78	0.45	0.17	63.07	63.06	3.02	0.23	20.00	0.00	0.00	0.00
0.9	10.65	5184.99	0.65	0.41	64.94	64.93	25.86	1.92	26.00	0.00	0.00	0.00
1.0	12.89	8277.83	0.86	1.13	65.56	65.55	43.37	8.03	33.00	0.00	0.00	0.03
25b2												
0.6	0.13	66.26	0.17	0.11	67.43	60.36	2.92	0.08	0.35	0.00	0.00	0.00
0.7	0.50	346.28	0.34	0.14	76.93	69.86	6.96	0.13	0.96	0.00	0.00	0.00
0.8	1.52	1006.19	0.83	0.19	85.45	78.38	31.64	0.26	2.70	0.00	0.00	0.05
0.9	3.54	2053.53	1.45	0.26	91.01	83.94	36.10	0.52	6.70	0.00	0.00	0.55
1.0	6.30	3330.55	1.85	0.45	94.01	86.94	58.93	1.02	12.00	0.00	0.00	1.34
25c2												
0.6	0.10	1.18	2195.12	2170.73	84.88	61.65	1.33	0.12	0.24	0.00	0.00	0.00
0.7	0.21	2.39	2333.33	2000.00	84.88	61.65	1.33	0.12	0.25	0.00	0.00	0.00
0.8	0.66	44.20	2377.36	2225.81	107.27	84.04	1.64	0.37	0.41	0.00	0.00	0.00
0.9	1.06	85.34	2401.61	2363.64	107.30	84.07	1.64	0.37	0.36	0.00	0.00	0.00
1.0	1.45	125.51	2363.35	2225.81	107.25	84.02	1.64	0.37	0.42	0.00	0.00	0.00
27a2												
0.8	0.50	1390.30	1.24	0.14	78.82	77.81	6.66	0.13	3.40	0.00	0.00	0.18
0.9	1.43	3506.43	1.86	0.25	79.90	78.89	36.56	0.40	13.00	0.00	0.00	0.59
1.0	2.59	6154.30	1.96	0.40	81.02	80.01	38.20	0.86	21.00	0.00	0.00	0.72
27b2												
0.8	0.19	104.43	0.27	0.12	83.51	80.48	5.73	0.09	0.39	0.00	0.00	0.00
0.9	1.09	556.57	1.07	0.14	92.15	89.12	9.01	0.20	1.80	0.00	0.00	0.12
1.0	3.05	1393.49	1.99	0.25	96.72	93.69	27.80	0.57	6.10	0.00	0.00	1.21
27c2												
0.8	0.10	1.18	2195.12	2170.73	84.88	81.85	1.33	0.12	0.24	0.00	0.00	0.00
0.9	0.21	2.39	2333.33	2000.00	84.88	81.85	1.33	0.12	0.25	0.00	0.00	0.00
1.0	0.66	44.20	2377.36	2225.81	107.27	104.24	1.64	0.37	0.41	0.00	0.00	0.00

ρ	D	CR	B_p	B_s	T	T_i	D_{ms}	D_{as}	LQ_p	LQ_s	LD_s	LD
22b2												
0.3	0.03	3.46	2166.67	2344.26	36.54	31.49	0.87	0.08	0.00	0.00	0.00	0.00
0.4	0.06	7.01	2157.33	2384.00	36.54	31.49	0.72	0.08	0.00	0.00	0.00	0.00
0.5	0.15	35.58	2327.41	2542.22	44.07	39.02	1.18	0.11	0.00	0.00	0.00	0.00
0.6	0.25	86.21	2693.88	2398.02	51.60	46.54	1.64	0.12	0.00	0.00	0.00	0.00
0.7	0.60	656.78	3431.56	2371.86	66.70	61.65	1.95	0.23	0.00	0.00	0.00	0.00
0.8	1.25	1583.96	7498.51	4369.60	77.86	72.81	5.17	0.43	0.00	0.00	0.00	0.00
0.9	2.15	2862.81	6994.90	19040.27	85.42	80.37	65.84	0.19	0.09	0.00	0.02	0.10
1.0	2.99	4037.35	6742.97	27575.63	88.93	83.88	96.67	0.19	0.13	0.00	0.02	0.09
22b4												
0.3	3.19	46.63	0.99	0.09	29.40	29.39	0.82	0.07	2.20	0.00	0.00	0.00
0.4	9.02	177.77	1.29	0.09	37.86	37.85	1.28	0.10	5.50	0.00	0.00	0.00
0.5	15.50	400.62	1.32	0.11	45.65	45.64	1.54	0.16	8.70	0.00	0.00	0.00
0.6	22.03	754.42	1.22	0.23	53.16	53.15	3.38	0.25	11.00	0.00	0.00	0.00
0.7	28.29	1286.73	1.10	0.29	60.37	60.36	6.40	0.46	14.00	0.00	0.00	0.00
0.8	34.48	2072.27	1.02	0.39	67.06	67.05	17.41	0.97	16.00	0.00	0.00	0.01
0.9	40.64	3139.62	1.04	1.59	72.64	72.63	65.79	3.84	19.00	0.00	0.00	0.05
1.0	47.01	4491.20	1.19	0.93	76.86	76.85	37.32	3.60	23.00	0.00	0.00	0.16
22c2												
0.3	0.00	0.00	0.00	0.00	12.30	7.25	0.00	0.00	0.00	0.00	0.00	0.00
0.4	0.00	0.00	0.00	0.00	12.30	7.25	0.00	0.00	0.00	0.00	0.00	0.00
0.5	0.00	0.00	0.00	0.00	34.89	29.84	0.00	0.00	0.00	0.00	0.00	0.00
0.6	0.00	0.00	0.00	0.00	34.89	29.84	0.00	0.00	0.00	0.00	0.00	0.00
0.7	0.00	0.00	0.00	0.00	34.89	29.84	0.00	0.00	0.00	0.00	0.00	0.00
0.8	0.00	0.00	0.00	0.00	34.89	29.84	0.00	0.00	0.00	0.00	0.00	0.00
0.9	0.00	0.00	0.00	0.00	46.18	41.13	0.00	0.00	0.00	0.00	0.00	0.00
1.0	0.00	0.00	0.00	0.00	46.18	41.13	0.00	0.00	20.00	0.00	0.00	0.00
23a2												
0.4	0.11	800.03	0.21	0.13	40.98	39.97	2.20	0.08	0.81	0.00	0.00	0.00
0.5	0.64	3697.49	0.52	0.18	48.71	47.70	20.17	0.17	5.00	0.00	0.00	0.00
0.6	1.80	8020.40	0.91	0.29	52.12	51.11	28.98	0.50	15.00	0.00	0.00	0.08
0.7	3.31	13130.24	1.09	0.47	54.13	53.12	50.59	0.87	26.00	0.00	0.05	0.20
0.8	4.97	22477.49	0.97	0.62	61.03	60.02	75.32	0.50	33.00	0.00	0.06	0.12
0.9	6.43	27330.00	1.16	1.02	59.57	58.56	76.90	1.32	43.00	0.00	0.06	0.24
1.0	8.06	31341.89	1.31	0.80	57.62	56.61	91.49	2.06	48.00	0.00	0.05	0.42

ρ	D	CR	B_p	B_s	T	T_j	D_{ms}	D_{as}	LQ_p	LQ_s	LD_s	LD
23a4												
0.4	1.48	213.81	0.59	0.10	36.95	36.94	0.31	0.06	6.90	0.00	0.00	0.00
0.5	3.80	1113.95	0.39	0.22	43.24	43.23	1.23	0.18	12.00	0.00	0.00	0.00
0.6	6.11	3375.62	0.43	0.19	49.05	49.04	1.59	0.21	17.00	0.00	0.00	0.00
0.7	8.40	7141.34	0.63	0.26	51.54	51.53	3.53	0.55	24.00	0.00	0.00	0.01
0.8	10.36	11359.73	0.87	0.87	52.72	52.71	29.80	2.19	33.00	0.00	0.00	0.05
0.9	12.52	15999.85	1.00	0.37	52.99	52.98	20.74	4.73	40.00	0.00	0.00	0.09
1.0	16.22	27091.06	1.08	3.44	59.45	59.44	13.31	3.01	48.00	0.00	6.25	0.16
23b2												
0.4	0.05	3.46	2177.11	2345.21	44.62	41.59	0.87	0.08	0.00	0.00	0.00	0.00
0.5	0.10	6.96	1840.54	2143.24	44.62	41.59	0.92	0.08	0.00	0.00	0.00	0.00
0.6	0.22	35.58	2330.29	2546.15	52.15	49.12	1.18	0.11	0.00	0.00	0.00	0.00
0.7	0.37	94.79	2131.93	2023.90	59.68	56.65	1.23	0.12	0.00	0.00	0.00	0.00
0.8	0.83	704.97	3692.56	2768.94	74.79	71.75	2.66	0.24	0.00	0.00	0.00	0.00
0.9	2.21	1974.98	11896.2	4423.96	85.74	82.71	9.78	0.57	0.00	0.00	0.00	0.07
1.0	2.70	3201.74	3774.43	23758.17	93.20	90.17	66.00	0.04	0.01	0.00	0.02	0.03
23c2												
0.3	0.00	0.00	0.00	0.00	12.30	7.25	0.00	0.00	0.00	0.00	0.00	0.00
0.4	0.00	0.00	0.00	0.00	12.30	7.25	0.00	0.00	0.00	0.00	0.00	0.00
0.5	0.00	0.00	0.00	0.00	34.89	29.84	0.00	0.00	0.00	0.00	0.00	0.00
0.6	0.00	0.00	0.00	0.00	34.89	29.84	0.00	0.00	0.00	0.00	0.00	0.00
0.7	0.00	0.00	0.00	0.00	34.89	29.84	0.00	0.00	0.00	0.00	0.00	0.00
0.8	0.00	0.00	0.00	0.00	34.89	29.84	0.00	0.00	0.00	0.00	0.00	0.00
0.9	0.00	0.00	0.00	0.00	46.18	41.13	0.00	0.00	0.00	0.00	0.00	0.00
1.0	0.00	0.00	0.00	0.00	46.18	41.13	0.00	0.00	20.00	0.00	0.00	0.00
25a2												
0.4	0.11	800.03	0.21	0.13	40.98	39.97	2.20	0.08	0.81	0.00	0.00	0.00
0.5	0.64	3697.49	0.52	0.18	48.71	47.70	20.17	0.17	5.00	0.00	0.00	0.00
0.6	1.80	8020.40	0.91	0.29	52.12	51.11	28.98	0.50	15.00	0.00	0.00	0.08
0.7	3.31	13130.24	1.09	0.47	54.13	53.12	50.59	0.87	26.00	0.00	0.05	0.20
0.8	4.97	22477.49	0.97	0.62	61.03	60.02	75.32	0.50	33.00	0.00	0.06	0.12
0.9	6.43	27330.00	1.16	1.02	59.57	58.56	76.90	1.32	43.00	0.00	0.06	0.24
1.0	8.06	31341.89	1.31	0.80	57.62	56.61	91.49	2.06	48.00	0.00	0.05	0.42

ρ	D	CR	B_p	B_s	T	T_i	D_{ms}	D_{as}	LQ_p	LQ_s	LD_s	LD
21a3												
0.2	0.20	385.14	0.27	0.14	19.85	19.75	2.05	0.08	1.20	0.00	0.00	0.00
0.3	0.70	1988.47	0.43	0.20	28.67	28.57	7.83	0.16	4.50	0.00	0.00	0.00
0.4	1.56	5401.15	0.56	0.18	35.60	35.50	13.67	0.50	11.00	0.00	0.00	0.00
0.5	2.97	10413.66	0.75	0.58	39.18	39.08	59.55	2.25	22.00	0.00	0.00	0.02
0.6	4.59	17457.75	0.86	1.60	42.84	42.74	41.22	0.99	32.00	0.00	0.66	0.06
0.7	6.82	22136.87	1.06	1.30	40.38	40.28	89.29	8.41	42.00	0.00	0.00	0.22
0.8	9.40	28485.85	1.15	0.72	40.35	40.25	62.57	7.45	49.00	0.00	0.00	0.34
0.9	12.17	35417.36	1.23	1.10	40.57	40.47	70.76	9.36	55.00	0.00	0.61	0.46
1.0	15.05	46284.95	1.23	1.66	43.36	43.26	59.08	1.02	62.00	0.00	0.67	0.43
21a4												
0.2	0.78	243.31	0.51	0.11	19.13	19.12	0.46	0.08	3.90	0.00	0.00	0.00
0.3	1.77	1356.89	0.36	0.19	27.75	27.74	2.25	0.22	7.10	0.00	0.00	0.00
0.4	2.85	4220.59	0.42	0.15	34.90	34.89	0.77	0.16	12.00	0.00	0.00	0.00
0.5	4.29	8988.02	0.64	0.42	39.06	39.05	10.50	1.88	21.00	0.00	0.00	0.01
0.6	6.21	14678.43	0.84	1.53	40.30	40.29	65.48	5.32	32.00	0.00	0.00	0.07
0.7	8.57	20532.72	1.01	0.75	39.95	39.94	52.38	11.99	42.00	0.00	0.00	0.17
0.8	11.11	27183.81	1.09	1.39	40.48	40.47	45.88	11.41	49.00	0.00	0.00	0.30
0.9	13.96	33704.72	1.17	0.53	40.22	40.21	37.68	12.23	55.00	0.00	0.00	0.33
1.0	16.96	40457.50	1.23	0.63	40.08	40.07	55.19	12.88	59.00	0.00	0.00	0.42
21b2												
0.2	0.03	13.90	0.12	0.12	27.18	20.11	2.36	0.08	0.30	0.00	0.00	0.00
0.3	0.11	80.56	0.16	0.13	37.01	29.94	5.07	0.11	0.62	0.00	0.00	0.00
0.4	0.27	265.78	0.21	0.18	46.80	39.73	18.48	0.18	1.00	0.00	0.00	0.00
0.5	0.56	646.44	0.29	0.23	56.25	49.18	21.30	0.28	1.80	0.00	0.00	0.00
0.6	1.10	1314.82	0.42	0.33	65.07	58.00	34.30	0.48	3.10	0.00	0.00	0.00
0.7	2.11	2372.19	0.63	0.44	72.93	65.86	59.65	0.76	5.30	0.00	0.00	0.01
0.8	3.86	3790.50	0.89	0.72	78.39	71.32	79.56	1.33	9.40	0.00	0.00	0.07
0.9	5.77	5172.64	0.93	1.42	79.99	72.92	55.76	0.07	11.00	0.00	0.02	0.08
1.0	8.64	7029.12	1.24	1.43	83.20	76.13	81.92	0.34	18.00	0.00	0.02	0.43
21c4												
0.2	1.42	10.42	0.45	0.12	19.96	19.87	3.02	0.13	0.54	0.00	0.00	0.00
0.3	4.00	46.25	0.82	0.12	29.75	29.66	3.43	0.22	1.30	0.00	0.00	0.00
0.4	7.23	118.04	1.10	0.14	39.36	39.27	6.30	0.34	2.40	0.00	0.00	0.00
0.5	10.98	241.71	1.24	0.15	48.54	48.45	6.55	0.42	3.60	0.00	0.00	0.00
0.6	15.26	438.81	1.30	0.32	57.42	57.33	18.64	0.83	4.90	0.00	0.00	0.02
0.7	20.24	732.98	1.42	0.68	65.56	65.47	44.65	1.53	6.70	0.00	0.00	0.11
0.8	25.78	1139.09	1.50	0.73	72.84	72.75	48.49	2.14	8.90	0.00	0.00	0.40
0.9	32.16	1677.53	1.65	1.33	79.22	79.13	91.85	4.70	12.00	0.00	0.00	0.84
1.0	37.40	2432.39	1.55	1.48	88.26	88.17	58.06	2.12	11.00	0.00	0.66	1.04

ρ	D	CR	B_p	B_s	T	T_i	D_{ms}	D_{as}	LQ_p	LQ_s	LD_s	LD
22a2												
0.3	0.08	587.53	0.18	0.14	31.01	30.00	3.79	0.08	0.66	0.00	0.00	0.00
0.4	0.41	2997.92	0.37	0.23	39.74	38.73	18.18	0.19	3.40	0.00	0.00	0.00
0.5	1.36	7413.04	0.71	0.34	44.81	43.80	49.92	0.46	13.00	0.00	0.00	0.02
0.6	2.85	12726.74	0.96	0.48	46.68	45.67	50.69	0.98	25.00	0.00	0.00	0.13
0.7	4.39	21081.35	0.92	0.80	52.08	51.07	53.09	0.66	33.00	0.00	0.06	0.10
0.8	6.16	25508.22	1.15	0.72	49.64	48.63	67.28	1.75	43.00	0.00	0.06	0.28
0.9	8.26	36177.52	1.09	0.57	53.78	52.77	58.27	0.14	51.00	0.00	0.07	0.23
1.0	10.21	43162.81	1.18	0.91	53.42	52.41	56.27	0.38	57.00	0.00	0.07	0.32
22a3												
0.3	0.45	532.08	0.43	0.12	29.37	29.27	1.02	0.08	2.40	0.00	0.00	0.00
0.4	1.43	2302.25	0.59	0.17	37.00	36.90	4.20	0.13	7.60	0.00	0.00	0.00
0.5	2.77	5593.72	0.68	0.29	42.70	42.60	13.67	0.57	15.00	0.00	0.00	0.00
0.6	4.48	10068.33	0.82	0.85	45.61	45.51	49.82	3.13	24.00	0.00	0.00	0.02
0.7	6.47	15233.50	0.96	0.66	46.83	46.73	52.63	4.46	33.00	0.00	0.00	0.10
0.8	8.64	20618.02	1.08	0.79	47.12	47.02	69.38	7.20	41.00	0.00	0.00	0.22
0.9	11.04	26381.77	1.18	1.35	47.27	47.17	74.91	10.36	48.00	0.00	0.00	0.27
1.0	13.56	32448.68	1.25	1.14	47.34	47.24	114.28	12.16	53.00	0.00	0.00	0.41
22a4												
0.3	1.24	231.45	0.57	0.11	28.06	28.05	0.56	0.08	5.90	0.00	0.00	0.00
0.4	3.01	1239.07	0.38	0.09	35.49	35.48	0.41	0.07	10.00	0.00	0.00	0.00
0.5	4.77	3807.36	0.43	0.32	42.09	42.08	6.40	0.85	15.00	0.00	0.00	0.00
0.6	6.61	8055.44	0.63	0.40	45.59	45.58	7.12	0.71	23.00	0.00	0.00	0.01
0.7	8.71	13006.50	0.87	1.20	46.28	46.27	42.44	6.65	33.00	0.00	0.00	0.06
0.8	10.97	18319.76	1.00	1.20	46.38	46.37	47.92	11.37	41.00	0.00	0.00	0.13
0.9	13.43	23993.05	1.10	0.84	46.45	46.44	52.17	8.60	47.00	0.00	0.00	0.20
1.0	16.05	29966.69	1.17	0.86	46.57	46.56	56.88	19.85	52.00	0.00	0.00	0.25

ρ	D	CR	B_p	B_s	T	T_i	D_{ms}	D_{as}	LQ_p	LQ_s	LD_s	LD
31a2												
0.2	0.04	285.56	0.12	0.07	19.89	18.88	0.00	0.00	0.88	0.00	0.00	0.00
0.3	0.21	1840.01	0.29	0.11	31.23	30.22	0.00	0.00	1.90	0.00	0.00	0.00
0.4	0.94	6223.26	0.53	0.32	38.67	37.66	0.20	0.00	11.00	0.00	0.00	0.03
0.5	2.58	11939.15	0.90	0.73	41.19	40.18	0.31	0.00	27.00	0.00	0.06	0.10
0.6	4.81	17777.92	1.08	1.01	41.10	40.09	0.41	0.00	37.00	0.00	0.31	0.25
0.7	7.45	24117.81	1.18	1.15	41.14	40.13	0.46	0.00	46.00	0.00	0.49	0.38
0.8	10.31	30254.57	1.25	1.13	40.59	39.58	0.46	0.00	52.00	0.00	0.30	0.52
0.9	13.51	36638.64	1.32	1.29	40.15	39.14	0.61	0.00	57.00	0.00	0.54	0.61
1.0	16.94	43717.09	1.36	1.31	40.23	39.22	0.46	0.00	62.00	0.00	1.19	0.63
31a3												
0.2	0.17	308.95	0.24	0.10	18.66	18.56	0.00	0.00	2.00	0.00	0.00	0.00
0.3	0.63	1795.25	0.41	0.10	29.54	29.44	0.00	0.00	4.00	0.00	0.00	0.00
0.4	1.58	5574.17	0.57	0.21	36.80	36.70	0.00	0.00	12.00	0.00	0.00	0.00
0.5	3.16	11012.88	0.84	0.54	41.16	41.06	0.00	0.00	25.00	0.00	0.00	0.04
0.6	5.16	16592.44	0.98	0.63	41.45	41.35	0.05	0.00	35.00	0.00	0.00	0.13
0.7	7.47	22603.38	1.10	0.95	41.31	41.21	0.05	0.00	44.00	0.00	0.00	0.31
0.8	10.04	28593.61	1.17	1.29	40.70	40.60	0.05	0.00	50.00	0.00	0.62	0.39
0.9	12.84	35110.68	1.25	0.69	40.52	40.42	0.05	0.00	56.00	0.00	0.61	0.52
1.0	15.79	41925.67	1.29	1.18	40.46	40.36	0.05	0.00	61.00	0.00	0.61	0.59
31a4												
0.2	0.74	161.89	0.57	0.00	18.01	18.00	0.00	0.00	4.40	0.00	0.00	0.00
0.3	1.78	1087.21	0.37	0.00	28.46	28.45	0.00	0.00	13.00	0.00	0.00	0.00
0.4	2.84	4402.32	0.36	0.24	36.34	36.33	0.00	0.00	25.00	0.00	0.00	0.00
0.5	4.41	9732.15	0.72	1.14	41.11	41.10	0.00	0.00	47.00	0.00	0.00	0.04
0.6	6.39	15264.83	0.91	0.69	41.08	41.07	0.00	0.00	75.00	0.00	0.00	0.11
0.7	8.74	21356.96	1.04	1.99	40.79	40.79	0.00	0.00	110.00	0.00	0.00	0.24
0.8	11.27	27407.31	1.12	0.52	40.56	40.55	0.00	0.00	140.00	0.00	0.00	0.31
0.9	14.11	33975.07	1.18	0.31	40.29	40.28	0.00	0.00	180.00	0.00	0.00	0.41
1.0	17.15	40706.38	1.25	2.26	40.04	40.03	0.00	0.00	220.00	0.00	0.00	0.47

ρ	D	CR	B_p	B_s	T	T_i	D_{ms}	D_{as}	LQ_p	LQ_s	LD_s	LD
31b2												
0.2	0.02	0.00	0.00	0.00	28.42	21.35	0.00	0.00	0.18	0.00	0.00	0.00
0.3	0.03	0.00	0.00	0.00	28.42	21.35	0.00	0.00	0.20	0.00	0.00	0.00
0.4	0.06	3.62	0.08	0.09	32.29	25.22	0.00	0.00	14.00	0.00	0.00	0.00
0.5	0.09	8.54	0.08	0.09	39.79	32.72	0.00	0.00	11.00	0.00	0.00	0.00
0.6	0.27	264.03	0.17	0.21	52.86	45.80	0.00	0.00	12.00	0.00	0.00	0.00
0.7	0.84	1077.07	0.40	0.33	64.50	57.43	0.00	0.00	9.00	0.00	0.00	0.00
0.8	1.68	2394.54	0.46	0.27	77.77	70.69	0.05	0.00	9.60	0.00	0.00	0.00
0.9	2.51	3573.99	0.45	0.25	80.99	73.93	0.05	0.00	9.80	0.00	0.00	0.00
1.0	5.20	5322.55	1.04	0.97	84.02	76.96	0.26	0.00	17.00	0.00	0.19	0.36
31c2												
0.2	0.05	0.00	0.00	0.00	36.57	21.44	0.00	0.00	0.00	0.00	0.00	0.00
0.3	0.09	0.00	0.00	0.00	36.54	21.40	0.00	0.00	0.00	0.00	0.00	0.00
0.4	0.18	0.00	0.00	0.00	59.09	43.97	0.00	0.00	0.00	0.00	0.00	0.00
0.5	0.26	0.00	0.00	0.00	59.14	44.04	0.00	0.00	0.00	0.00	0.00	0.00
0.6	0.34	0.00	0.00	0.00	59.12	43.99	0.00	0.00	0.00	0.00	0.00	0.00
0.7	0.43	0.00	0.00	0.00	59.12	43.99	0.00	0.00	0.00	0.00	0.00	0.00
0.8	0.53	0.00	0.00	0.00	70.46	55.29	0.00	0.00	0.00	0.00	0.00	0.00
0.9	0.63	4.54	0.13	0.24	70.66	55.58	0.00	0.00	20.00	0.00	0.00	0.00
1.0	0.73	7.94	0.12	0.10	70.66	55.58	0.00	0.00	20.00	0.00	0.00	0.00
32a2												
0.3	0.06	449.37	0.14	0.13	29.79	28.78	0.00	0.00	1.20	0.00	0.00	0.00
0.4	0.38	2894.10	0.35	0.19	40.71	39.70	0.00	0.00	3.00	0.00	0.00	0.00
0.5	1.49	7494.99	0.74	0.51	45.48	44.47	0.20	0.00	15.00	0.00	0.00	0.02
0.6	3.33	12718.29	1.09	0.84	46.98	45.98	0.36	0.00	29.00	0.00	0.16	0.19
0.7	5.47	18040.23	1.22	0.98	47.12	46.11	0.41	0.00	37.00	0.00	0.48	0.35
0.8	7.92	23624.42	1.32	1.07	47.05	46.04	0.41	0.00	45.00	0.00	0.74	0.48
0.9	10.62	29425.09	1.37	1.27	47.03	46.02	0.41	0.00	50.00	0.00	1.12	0.56
1.0	13.42	35196.05	1.43	1.18	46.82	45.82	0.46	0.00	55.00	0.00	0.74	0.63
33a2												
0.4	0.09	709.50	0.19	0.00	39.86	38.85	0.00	0.00	1.00	0.00	0.00	0.00
0.5	0.57	3711.55	0.44	0.19	49.77	48.76	0.00	0.00	4.40	0.00	0.00	0.00
0.6	1.88	7959.82	0.98	0.50	52.20	51.19	0.20	0.00	18.00	0.00	0.00	0.13
0.7	3.72	12631.79	1.27	0.84	53.22	52.21	0.26	0.00	30.00	0.00	0.14	0.30
0.8	5.79	17475.39	1.37	0.93	53.62	52.61	0.36	0.00	37.00	0.00	0.51	0.43
0.9	7.99	22277.24	1.46	1.02	53.52	52.51	0.41	0.00	44.00	0.00	0.41	0.60
1.0	10.35	27368.85	1.51	1.08	53.59	52.58	0.46	0.00	48.00	0.00	0.69	0.65

ρ	D	CR	B_p	B_s	T	T_i	D_{ms}	D_{as}	LQ_p	LQ_s	LD_s	LD
35a2												
0.6	0.15	970.22	0.26	0.08	59.53	58.52	0.00	0.00	1.50	0.00	0.00	0.00
0.7	1.10	4148.29	1.02	0.27	65.41	64.40	0.15	0.00	9.40	0.00	0.00	0.14
0.8	2.51	7350.39	1.46	0.56	65.92	64.91	0.26	0.00	21.00	0.00	0.06	0.40
0.9	4.18	10876.39	1.65	0.84	66.52	65.51	0.31	0.00	31.00	0.00	0.48	0.61
1.0	5.91	14271.14	1.75	1.05	66.55	65.54	0.41	0.00	37.00	0.00	0.89	0.72

APPENDIX B PROTOCOL SIMULATOR PROGRAMS

B.1 ETHERNET PROTOCOL SIMULATOR

This program is an Ethernet simulator with Poisson offered loads.

```
program ENet_sim(input,eao);
 function uniform(a,b:real;var U:integer):real; extern;
 const
  gap = 10;              {distance, in meters, between terminals}
  packet_length = 1;     {length of packet in slot-times}
  slot_time = 51.2e-6;   {round-trip propagation delay, in seconds}
  N = 40;                {number of terminals}
  qlength_p = 1;    {number of poisson generated packets that can be}
                    {stored by a terminal}
  qlength_s = 5;    {number of scheduled packets that can be}
                    {stored by a terminal}
  ST = 50000;            {number of successful transmissions that must be}
                    {completed for the simulation}
  pi = 3.141592654;
{**********************************************************************
**********}
 type
  ptr = ^ node;
  packet = record
    length:integer;          {number of slot times required to transmit re- }
                        {mainder of the packet.}
    delay:integer;           {number of slot times between generation of }
                        {packet and transmission}
  end;
  data = array[1..qlength_s] of packet;
  node = record                {Each node in linked lists contains information}
                        {describing one terminal on the network.}
    transmitting:boolean;     {transmitting flag (is terminal transmitting)}
    queue:data;              {info. describing queued pack-}
                        {ets}
    backoff:integer;         {slot-times until terminal may again attempt}
                        {transmission}
    attempts:integer;        {number of attempts at transmission of packet}
                        {described by queue[1] }
    distance:integer;        {distance to previous terminal}
```

```
  threshold:real;          {used to determine if terminal has data arri-}
                          {ving during a slot time}
  next:ptr;                {pointer to next node on network}
 end;
{**************************************************************************
**********}
 var
 eao:text;
 begin_delay:integer;        {sum of all delays experienced by transmitted}
                          {packets}
 collisions:integer;         {number of collisions}
 rho:real;                {normalized throughput of the network}
 steps:array[1..15] of real;  {values of rho}
 channel_busy,busy:boolean;   {is the channel being used?}
 etime:integer;              {number of slot times simulated}
 transmissions:integer;       {number of successful transmissions completed}
 transmitters: integer;       {number of terminals with data to transmit in a}
                          {given slot time}
 terminal:ptr;               {pointer to a terminal}
 first:ptr;                {pointer to the first terminal}
 packets_lost:integer;        {total number of packets lost due}
                          {to excessive delay}
 q_lost_p:integer;        {poisson generated and scheduled packets lost}
                          {due to insufficient queue space}
 SEEDi:integer;              {seeds for random number generators}
 i,j,ind,z:integer;
 x:real;
 poisson_packets:integer; {tells how many of each type}
                          {of packet was transmitted}
 p_count:integer;         {number of scheduled and poisson type packets}
                          {generated}
 bkoffs,bkftime:integer;  {number of times terminals go into backoff, }
                        {and total amount of time spent in backoff}
{**************************************************************************
**********}
 function realint(n:integer):real;
  begin
   realint:=n;
  end;
{**************************************************************************
**********}
 procedure initialize(var first:ptr:N,qlength:integer:rho:real;
                                              var SEEDi:integer);
```

```
var
 i,j:integer;
 terminal:ptr;
 t:real;
begin
 SEEDi:=56912;
 t:=rho/N/packet_length;
 new(first);
 terminal:=first;
 for i:=1 to N do
  begin
   with terminal^ do
    begin
     transmitting:=false;
     distance:=gap;
     attempts:=0;
     backoff:=0;
     threshold:=t;
     for j:=1 to qlength do
      with queue[j] do
       begin
        length:=0;
        delay:=0
       end
    end;
   if (i=N) then
    terminal^.next:=nil
   else
    begin
     new(terminal^.next);
     terminal:=terminal^.next;
    end
  end
 end;
{**********************************************************************
**********}
 procedure randint(var SEED,n:integer;limit:integer);
{This procedure produces a random integer in the variable n where 0<n<limit.}
 begin
  n:=1+round(realint(limit-1)*uniform(0,1,SEED));
 end;
{**********************************************************************
**********}
```

```
procedure inbackoff(var terminal:node; var SEED,packets_lost,bkoffs,bkftime:inte-
ger);
 var
  i:integer;
 begin
  bkoffs:=bkoffs+1;
  with terminal do
   begin
    attempts:=attempts+1;
    if attempts>16 then
     begin
      attempts:=0;
      packets_lost:=packets_lost+1;
      for i:=1 to (qlength_s-1) do
       queue[i]:=queue[i+1];
      with queue[qlength_s] do
       begin
        length:=0;
        delay:=0
       end
     end
    else
     if attempts<=8 then
      randint(SEED,backoff,round(exp(attempts*ln(2))))
     else
      randint(SEED,backoff,256);
     bkftime:=bkftime+backoff
   end
 end;
{**************************************************************************
**********}
 procedure add_data(var queue:data;packet_size:integer;var lost:integer);
 var
  i,qlength:integer;
 begin
  qlength:=qlength_p;
  i:=1;                    {move to empty spot in queue}
  if qlength>1 then
   while  (i<qlength) and(queue[i].length>0) do
    i:=i+1;
  if queue[i].length=0 then     {i is at empty spot in queue, or}
   with queue[i] do            {at the lost spot in queue}
    begin
```


C-5

```
    length:=packet_size;
    delay:=-1
   end
  else
   lost:=lost+1
 end;                          {end procedure}
{**********************************************************************
**********}
 begin                         {begin main program}
  rewrite(eao);
  writeln(eao,'N = number of stations = ',N);
  writeln(eao,'packet length=',packet_length:0);
  writeln(eao,'# of transmissions that must be completed is',ST);
  writeln(eao,'This is the Ethernet simulator.');
  steps[1]:=0.1;
  steps[2]:=0.2;
  steps[3]:=0.3;
  steps[4]:=0.4;
  steps[5]:=0.5;
  steps[6]:=0.6;
  steps[7]:=0.7;
  steps[8]:=0.8;
  steps[9]:=0.9;
  steps[10]:=1.0;
  steps[11]:=3.0;
  steps[12]:=5.0;
  steps[13]:=7.0;
  steps[14]:=10.0;
  for z:=1 to 14 do
   begin
    rho:=steps[z];
    writeln(eao);
    writeln(eao,'rho =',rho:3:1);
    channel_busy:=false;
    transmissions:=0;   {init stats for each run}
    bkftime:=0;
    bkoffs:=0;
    q_lost_p:=0;
    packets_lost:=0;
    etime:=0;
  initialize(first,N,qlength_s,rho,SEEDi);
  while transmissions<ST do
   begin
```

```
terminal:=first;
ind:=0;
transmitters:=0;
busy:=channel_busy;
while terminal<>nil do
 begin
  ind:=ind+1;
  with terminal^ do
   begin
    if backoff>0 then backoff:=backoff-1;
    x:= uniform(0,1,SEEDi);
    if abs(x)<threshold then    {is there new data acquired during}
                                {this slot time?}
     begin
      p_count:=p_count+1;
      add_data(queue,packet_length,q_lost_p)
     end;
    i:=1;
    while  (i<=qlength_s) and (queue[i].length>0) do
    with queue[i] do
     begin
      if i=1 then
       begin
        if transmitting then
         begin
          length:=length-1;
          if length=0 then {has all of packet been transmitted?}
           begin
            busy:=false;
            transmitting:=false;
            transmissions:=transmissions+1;
            for j:=1 to qlength_s-1 do
             queue[j]:=queue[j+1];
            queue[qlength_p].length:=0;
           end;          {end if all of packet transmitted}
         end             {end if transmitting}
       end;              {end if i=1}
      delay:=delay+1;
      i:=i+1;
     end;               {end with queue[i]}
    terminal:=next;
   end;                 {end with terminal}
 end;                       {end while terminal<>nil}
```

```
   channel_busy:=busy;
terminal:=first;
ind:=0;
while terminal<>nil do
 begin
  with terminal^ do
           begin
        if(queue[1].length>0) and (queue[1].length>0) and (not channel_busy) and
(backoff=0) then
                          {would this terminal like to transmit?}
             begin
              transmitters:=transmitters+1;
              if transmitters>1 then
               inbackoff(terminal^,SEEDi,packets_lost,bkoffs,bkftime);
             end;            {end if this terminal has data}
  terminal:=terminal^.next;
end;
           end;
    etime:=etime+1;
    if (not channel_busy) and (transmitters>0) then
     begin
      terminal:=first;
      while (terminal^.queue[1].length=0) or (terminal^.backoff>0) do
       terminal:=terminal^.next;
      with terminal^ do
       begin
        if transmitters=1 then
         begin
          channel_busy:=true;
          transmitting:=true;
          attempts:=0;
          backoff:=0;
          begin_delay:=begin_delay+queue[1].delay
         end
        else
         begin
          collisions:=collisions+1;
          inbackoff(terminal^,SEEDi,packets_lost,bkoffs,bkftime)
         end
       end;
     end;          {end with terminal^}
   end;            {ST transmissions have now been simulated}
  write(eao,'Mean Delay = ',realint(begin_delay)/ST*slot_time:7:5,' seconds');
```

```
  writeln(eao,' = ',realint(begin_delay)/ST:7:5,' slot times');
  write(eao,'Mean backoff time = ',realint(bkftime)/realint(bkoffs)*slot_time:7:5,'
seconds');
  writeln(eao,' = ',realint(bkftime)/realint(bkoffs):7:5,' slot times');
  write(eao,'Collision Rate = ',realint(collisions)/realint(etime)/slot_time:7:5);
  writeln(eao,' collisions/second');
  write(eao,'Packets lost due to excessive delay = ');
  writeln(eao,realint(packets_lost)/ST*100:7:5,'%');
  writeln(eao,etime:0,' slot times were simulated.');
  writeln(eao,'Throughput = ',ST*packet_length/realint(etime)*100:7:5,'%');
  write(eao,'Poisson packets lost due to insufficient queue space = ');
  write(eao,q_lost_p:0,' = ',realint(q_lost_p)/realint(p_count)*100:7:5);
  writeln(eao,'%');
 end;
 close(eao);
end.
```

B.2 MFA SIMULATOR PROGRAM ONE

This program is for scheduled data using the Ethernet protocol.

```
program MNet_sim(input,pro11a2);
{This program simulates and Ethernet network where for there is scheduled data}
{based on a cycle of length cycle_length. When a terminal acquires scheduled }
{data, it must contend for the network as though the data were normal, poisson}
{data.}
 function uniform(a,b:real;var U:integer):real; extern;
 const
  gap=10;              {distance, in meters, between terminals}
  packet_length=1;     {length of packet in slot-times}
  slot_time=51.2e-6;   {round-trip propagation delay, in seconds}
  N=40;                {number of terminals}
  qlength_p=1;    {number of poisson generated packets that can be}
                  {stored by a terminal}
  qlength_s=5;    {number of scheduled packets that can be}
                  {stored by a terminal}
  ST=50000;            {number of successful transmissions that must be}
                  {completed for the simulation}
  pi=3.141592654;
  cycle_length=100;   {length of MFA cycle in slot-times}
{**********************************************************************
**********}
 type
  ptr=^node;
```

```
how_generated=(scheduled,poisson);
 bkfcount=array[how_generated] of integer;
packet=record
  length:integer;          {number of slot times required to transmit re- }
                          {mainder of the packet.}
  delay:integer;           {number of slot times between generation of }
                          {packet and transmission}
  data_type:how_generated   {indicates whether the packet is scheduled,}
                          {or poisson-generated data}
end;
data=array[1..qlength_s] of packet;
node=record                {Each node in linked lists contains information}
                          {describing one terminal on the network.}
  transmitting:boolean;     {transmitting flag (is terminal transmitting)}
  queue:data;              {info. describing queued pack-}
                          {ets}
  backoff:integer;          {slot-times until terminal may again attempt}
                          {transmission}
  attempts:integer;         {number of attempts at transmission of packet}
                          {described by queue[1] }
  distance:integer;         {distance to previous terminal}
  threshold:real;          {used to determine if terminal has data arri-}
                          {ving during a slot time}
  next:ptr;                {pointer to next node on network}
end;
{**************************************************************************
**********}
var
 pro11a2:text;
 begin_delay:integer;       {sum of all delays experienced by transmitted}
                          {packets}
 collisions:integer;        {number of collisions}
 rho:real;                 {normalized throughput of the network}
 steps:array[1..12] of real;  {values of rho}
 channel_busy,busy:boolean;   {is the channel being used?}
 etime:integer;            {number of slot times simulated}
 mtime:integer;            {mfa cycle time counter}
 transmissions:integer;     {number of successful transmissions completed}
 transmitters: integer;     {number of terminals with data to transmit in a}
                          {given slot time}
 terminal:ptr;             {pointer to a terminal}
 first:ptr;                {pointer to the first terminal}
 packets_lost:integer;      {total number of packets lost due}
```

```
                    {to excessive delay}
 q_lost_p,q_lost_s:integer;     {poisson generated and scheduled packets lost}
                    {due to insufficient queue space}
 SEEDi:integer;              {seeds for random number generators}
 i,j,ind,z:integer;
 x:real;
 fraction_scheduled:real;     {ratio of scheduled traffic to bandwidth}
 scheduled_packets,poisson_packets:integer; {tells how many of each type}
                    {of packet was transmitted}
 s_count,p_count:integer;     {number of scheduled and poisson type packets}
                    {generated}
 cycle_count:integer;         {number of scheduled packets that must be gen-}
                    {erated per cycle}
 sc_count:integer;            {count of scheduled packets generated in curr-}
                    {ent cycle}
 bkoffs,bkftime:bkfcount;  {number of times terminals go into backoff, }
                  {and total amount of time spent in backoff}
{**********************************************************************
**********}
 function realint(n:integer):real;
  begin
   realint:=n;
  end;
{**********************************************************************
**********}
 procedure initialize(var first:ptr;N,qlength:integer;rho:real;
                                          var SEEDi:integer);
  var
   i,j:integer;
   terminal:ptr;
   t:real;
  begin
   SEEDi:=56912;
   t:=(rho-fraction_scheduled)/N/packet_length;
   new(first);
   terminal:=first;
   for i:=1 to N do
    begin
     with terminal^ do
      begin
       transmitting:=false;
       distance:=gap;
       attempts:=0;
```

```
      backoff:=0;
      threshold:=t;
      for j:=1 to qlength do
       with queue[j] do
        begin
         length:=0;
         delay:=0
        end
      end;
     if (i=N) then
      terminal^.next:=nil
     else
      begin
       new(terminal^.next);
       terminal:=terminal^.next;
      end
    end
  end;
{**********************************************************************
**********}
 procedure randint(var SEED,n:integer;limit:integer);
{This procedure produces a random integer in the variable n where 0<n<limit.}
  begin
   n:=1+round(realint(limit-1)*uniform(0,1,SEED));
  end;
{**********************************************************************
**********}
   procedure inbackoff(var terminal:node; var SEED,packets_lost:integer; var
bkoffs,bkftime:bkfcount);
  var
   i:integer;
  begin
   with terminal do
    begin
     attempts:=attempts+1;
     if attempts>16 then
      begin
       attempts:=0;
       packets_lost:=packets_lost+1;
       for i:=1 to (qlength_s-1) do
        queue[i]:=queue[i+1];
       with queue[qlength_s] do
        begin
```

```
        length:=0;
        delay:=0
       end
     end
    else
     begin
     bkoffs[queue[1].data_type]:=bkoffs[queue[1].data_type]+1;
     if attempts<=8 then
      randint(SEED,backoff,round(exp(attempts*ln(2))))
     else
      randint(SEED,backoff,256);
     bkftime[queue[1].data_type]:=bkftime[queue[1].data_type]+backoff
     end
   end
  end;
{**********************************************************************
**********}
 procedure add_data(var queue:data;packet_size:integer;which:how_generated;var
lost:integer);
 var
  i,qlength:integer;
 begin
  case which of
   scheduled: qlength:=qlength_s;
   poisson:  qlength:=qlength_p
  end;   {end case}
  i:=1;                    {move to empty spot in queue}
  if qlength>1 then
   while  (i<qlength) and(queue[i].length>0) do
    i:=i+1;
  if queue[i].length=0 then     {i is at empty spot in queue, or}
   with queue[i] do           {at the lost spot in queue}
    begin
     length:=packet_size;
     data_type:=which;
     delay:=-1
    end
  else
   lost:=lost+1
 end;                     {end procedure}
{**********************************************************************
**********}
 begin                    {begin main program}
```

```
 rewrite(pro11a2);
 write(pro11a2,'These are the results of a simulation of a network in which');
 write(pro11a2,' each terminal acquires scheduled data during a ');
 write(pro11a2,'cycle of length ',cycle_length:0,' slot times such that');
 write(pro11a2,' terminal 1 receives data at the beginning of the cycle, ');
 write(pro11a2,'terminal 2 receives data 1 packet length later, and so on,');
 write(pro11a2,' until all the scheduled data is accounted for. The remain');
 write(pro11a2,'ing time in the cycle is used by the network for transmission');
 write(pro11a2,' of poisson generated traffic according to an Ethernet scheme.');
 write(pro11a2,' When a terminal receives scheduled data it must contend');
 writeln(pro11a2,' for the channel.');
 writeln(pro11a2);
 writeln(pro11a2,'N = number of stations = ',N);
writeln(pro11a2,'packet length = ',packet_length:0,' cycle length = ',cycle_length:0);
 writeln(pro11a2,'# of transmissions that must be completed is',ST);
 fraction_scheduled:=0.1;
 writeln(pro11a2,'fraction of scheduled traffic = ',fraction_scheduled:3:1);
 writeln(pro11a2,'This is program 1.');
 steps[1]:=0.1;
 steps[2]:=0.2;
 steps[3]:=0.3;
 steps[4]:=0.4;
 steps[5]:=0.5;
 steps[6]:=0.6;
 steps[7]:=0.7;
 steps[8]:=0.8;
 steps[9]:=0.9;
 steps[10]:=1.0;
 cycle_count:=trunc(fraction_scheduled*cycle_length/packet_length);
  if realint(trunc(fraction_scheduled*cycle_length/packet_length))<fraction_sched-
uled*cycle_length/packet_length then
  cycle_count:=cycle_count+1;
 for z:=1 to 10 do
  begin
   rho:=steps[z];
   writeln(pro11a2);
   writeln(pro11a2,'rho = ',rho:3:1);
   channel_busy:=false;
if rho>fraction_scheduled then
 begin
   transmissions:=0;  {init stats for each run}
   bkftime[scheduled]:=0;
   bkftime[poisson]:=0;
```

```
   bkoffs[scheduled]:=0;
   bkoffs[poisson]:=0;
   s_count:=0;
   q_lost_s:=0;
   q_lost_p:=0;
   packets_lost:=0;
   etime:=0;
   mtime:=0;
   sc_count:=0;
  initialize(first,N,qlength_s,rho,SEEDi);
  while transmissions<ST do
   begin
   terminal:=first;
   ind:=0;
   transmitters:=0;
   busy:=channel_busy;
   while terminal<>nil do
    begin
     ind:=ind+1;
     with terminal^ do
      begin
       if backoff>0 then backoff:=backoff-1;
     if ((mtime) mod (N) = (ind-1)*packet_length) and (sc_count<cycle_count) then
                                          {beginning of MFA cycle}
begin
        sc_count:=sc_count+1;
        s_count:=s_count+1;
        add_data(queue,packet_length,scheduled,q_lost_s)
       end
      else
       begin
         x:= uniform(0,1,SEEDi);
        if abs(x)<threshold then    {is there new data acquired during}
                                    {this slot time?}
         begin
          p_count:=p_count+1;
          add_data(queue,packet_length,poisson,q_lost_p)
         end
       end;
      i:=1;
      while  (i<=qlength_s) and (queue[i].length>0) do
      with queue[i] do
       begin
```

```
        if i=1 then
         begin
          if transmitting then
           begin
            length:=length-1;
            if length=0 then {has all of packet been transmitted?}
             begin
              busy:=false;
              transmitting:=false;
              transmissions:=transmissions+1;
              for j:=1 to qlength_s-1 do
               queue[j]:=queue[j+1];
              queue[qlength_s].length:=0;
             end;          {end if all of packet transmitted}
           end             {end if transmitting}
         end;              {end if i=1}
         delay:=delay+1;
         i:=i+1;
       end;                {end with queue[i]}
      terminal:=next;
     end;                 {end with terminal}
   end;                       {end while terminal<>nil}
  channel_busy:=busy;
terminal:=first;
ind:=0;
while terminal<>nil do
 begin
  with terminal^ do
          begin
           if (queue[1].length>0) and  (not channel_busy) and (backoff=0) then
                        {would this terminal like to transmit?}
            begin
             transmitters:=transmitters+1;
             if transmitters>1 then
              inbackoff(terminal^.SEEDi.packets_lost.bkoffs.bkftime);
            end;          {end if this terminal has data}
  terminal:=terminal^.next;
end;
          end;
   etime:=etime+1;
   mtime:=mtime+1;              {setting cycle counters}
      if mtime=cycle_length then
       begin
```

```
        sc_count:=0;
        mtime:=0;
       end;
   if (not channel_busy) and (transmitters>0) then
    begin
     terminal:=first;
     while (terminal^.queue[1].length=0) or (terminal^.backoff>0) do
      terminal:=terminal^.next;
     with terminal^ do
      begin
       if transmitters=1 then
        begin
         channel_busy:=true;
         transmitting:=true;
         attempts:=0;
         backoff:=0;
         begin_delay:=begin_delay+queue[1].delay
        end
       else
        begin
         collisions:=collisions+1;
         inbackoff(terminal^,SEEDi,packets_lost,bkoffs,bkftime)
        end
      end;
    end;          {end with terminal^}
  end;            {ST transmissions have now been simulated}
 write(pro11a2,'Mean Delay = ',realint(begin_delay)/ST*slot_time:7:5,' seconds');
 writeln(pro11a2,' = ',realint(begin_delay)/ST:7:5,' slot times');
 write(pro11a2,'Collision Rate = ',realint(collisions)/realint(etime)/slot_time:7:5);
 writeln(pro11a2,' collisions/second');
  write(pro11a2,'Mean poisson backoff time = ',realint(bkftime[poisson])/real-
int(bkoffs[poisson])*slot_time:7:5,' seconds');
  writeln(pro11a2,' = ',realint(bkftime[poisson])/realint(bkoffs[poisson]):7:5,' slot
times');
 write(pro11a2,'Mean scheduled backoff time = ',realint(bkftime[scheduled])/real-
int(bkoffs[scheduled])*slot_time:7:5,' seconds');
  writeln(pro11a2,' = ',realint(bkftime[scheduled])/realint(bkoffs[scheduled]):7:5,'
slot times');
 write(pro11a2,'Packets lost due to excessive delay = ');
 writeln(pro11a2,realint(packets_lost)/ST*100:7:5,'%');
 writeln(pro11a2,etime:0,' slot times were simulated.');
 writeln(pro11a2,'Throughput = ',ST*packet_length/realint(etime)*100:7:5,'%');
 writeln(pro11a2,'Scheduled packets generated = ',s_count:0);
```

```
  write(pro11a2,'Scheduled packets lost due to insufficient queue space = ');
  write(pro11a2,q_lost_s:0,' = ',realint(q_lost_s)/realint(s_count)*100:7:5);
  writeln(pro11a2,'%');
  write(pro11a2,'Poisson packets lost due to insufficient queue space = ');
  write(pro11a2,q_lost_p:0,' = ',realint(q_lost_p)/realint(p_count)*100:7:5);
  writeln(pro11a2,'%');
 end;
end;
 close(pro11a2);
end.
```

B.3 MFA SIMULATOR PROGRAM TWO

This program implements the MFA protocol but the sync packet must contend for the channel.

```
program MNet_sim(input,pro21a2);
{This program simulates and Ethernet network where for there is scheduled data}
{based on a cycle of length cycle_length.  When a terminal acquires scheduled }
{data, it must contend for the network as though the data were normal, poisson}
{data.}
 function uniform(a,b:real;var U:integer):real; extern;
 const
  gap = 10;              {distance, in meters, between terminals}
  packet_length = 1;     {length of packet in slot-times}
  slot_time = 51.2e-6;   {round-trip propagation delay, in seconds}
  N = 40;                {number of terminals}
  qlength_p = 1;    {number of poisson generated packets that can be}
                    {stored by a terminal}
  qlength_s = 5;    {number of scheduled packets that can be}
                    {stored by a terminal}
  ST = 50000;            {number of successful transmissions that must be}
                    {completed for the simulation}
  pi = 3.141592654;
  cycle_length = 100;   {length of MFA cycle in slot-times}
{**********************************************************************
**********}
 type
  how_generated = (scheduled,poisson);
   bkfcount = array[how_generated] of integer;
  ptr = ^ node;
  packet = record
```

```
  length:integer;          {number of slot times required to transmit re- }
                          {mainder of the packet.}
  delay:integer;           {number of slot times between generation of }
                          {packet and transmission}
  data_type:how_generated   {indicates whether the packet is scheduled.}
                          {or poisson-generated data}
 end;
 data = array[1..qlength_s] of packet;
 node = record              {Each node in linked lists contains information}
                          {describing one terminal on the network.}
  transmitting:boolean;     {transmitting flag (is terminal transmitting)}
  queue:data;              {info. describing queued pack-}
                          {ets}
  backoff:integer;          {slot-times until terminal may again attempt}
                          {transmission}
  attempts:integer;         {number of attempts at transmission of packet}
                          {described by queue[1] }
  distance:integer;         {distance to previous terminal}
  threshold:real;          {used to determine if terminal has data arri-}
                          {ving during a slot time}
  next:ptr;                {pointer to next node on network}
 end;
{**************************************************************************
**********}
 var
 pro21a2:text;
 begin_delay:integer;       {sum of all delays experienced by transmitted}
                          {packets}
 collisions:integer;        {number of collisions}
 rho:real;                 {normalized throughput of the network}
 steps:array[1..12] of real;  {values of rho}
 channel_busy,busy:boolean;   {is the channel being used?}
 etime:integer;            {number of slot times simulated}
 mtime:integer;             {mfa cycle time counter}
 transmissions:integer;     {number of successful transmissions completed}
 transmitters: integer;     {number of terminals with data to transmit in a}
                          {given slot time}
 terminal:ptr;             {pointer to a terminal}
 first:ptr;                {pointer to the first terminal}
 pkslst_s,pkslst_p:integer;  {total number of scheduled and poisson}
                          {packets lost due to excessive delay}
 q_lost_p,q_lost_s:integer;      {total number of packets lost due to insufficient-}
                          {queue space}
```

```
 packets_generated:integer;    {total number of packets generated}
 SEEDi:integer;                {seeds for random number generators}
 i,j,ind,z:integer;
 x:real;
 fraction_scheduled:real;      {ratio of scheduled traffic to bandwidth}
 scheduled_data:boolean;       {flag indicating the beginning of a cycle}
lp_in_queue:boolean;     {flag indicating scheduled data has been placed in a queue}
 mdelay_max,mdelay,mdelay_total:integer;    {maximum, delay during cycle and }
                        {total delay experienced by scheduled data}
                        {(in slot times)}
 cycles:integer;              {number of cycles simulated}
 long_packet:integer;         {the length of the scheduled, extended packet}
 infotxms:real;   {number of packet of useful information transmitted}
 bkoffs,bkftime:bkfcount;   {number of times terminals go into backoff, }
                      {and total amount of time spent in backoff}
{**************************************************************************
**********}
 function realint(n:integer):real;
  begin
   realint:=n;
  end;
{**************************************************************************
**********}
 procedure initialize(var first:ptr;N,qlength:integer;rho:real;
                                              var SEEDi:integer);
  var
   i,j:integer;
   terminal:ptr;
   t:real;
  begin
   SEEDi:=56912;
   t:=(rho-fraction_scheduled)/N/packet_length;
   new(first);
   terminal:=first;
   for i:=1 to N do
    begin
     with terminal^ do
      begin
       transmitting:=false;
       distance:=gap;
       attempts:=0;
       backoff:=0;
       threshold:=t;
```

```
      for j:=1 to qlength do
       with queue[j] do
        begin
         length:=0;
         delay:=0
        end
      end;
     if (i=N) then
      terminal^.next:=nil
     else
      begin
       new(terminal^.next);
       terminal:=terminal^.next;
      end
    end
  end;
{*************************************************************************
**********}
 procedure randint(var SEED,n:integer;limit:integer);
{This procedure produces a random integer in the variable n where 0<n<limit.}
  begin
   n:=1+round(realint(limit-1)*uniform(0,1,SEED));
  end;
{*************************************************************************
**********}
  procedure inbackoff(var terminal:node; var SEED,pkslst_s,pkslst_p:integer; var
bkoffs,bkftime:bkfcount);
  var
   i:integer;
  begin
   with terminal do
    begin
     attempts:=attempts+1;
     if attempts>16 then
      begin
       attempts:=0;
       case queue[1].data_type of
        poisson: pkslst_p:=pkslst_p+1;
        scheduled: pkslst_s:=pkslst_s+1;
       end;  {end case}
       for i:=1 to (qlength_s-1) do
        queue[i]:=queue[i+1];
       with queue[qlength_s] do
```

```
        begin
         length:=0;
         delay:=0
        end
       end
      else
       begin
       bkoffs[queue[1].data_type]:=bkoffs[queue[1].data_type]+1;
       if attempts<=8 then
        randint(SEED,backoff,round(exp(attempts*ln(2))))
       else
        randint(SEED,backoff,256);
       bkftime[queue[1].data_type]:=bkftime[queue[1].data_type]+backoff
       end
     end
  end;
{**********************************************************************
**********}
 function empty(queue:data;qlength:integer):integer;
{This function checks to see if there is an empty spot on the queue, and if}
{so, returns the place of that spot.  If there is no empty spot, the function}
{returns a zero.}
  var
   i:integer;
  begin
   i:=1;
   if qlength>1 then
     while  (i<qlength) and(queue[i].length>0) do
      i:=i+1;
   if queue[i].length=0 then
    empty:=i
   else
    empty:=0;
  end;
{**********************************************************************
**********}
  procedure  add_data(var  queue:data;packet_size:integer;which:how_generated;var
lost:integer);
 var
  i,qlength:integer;
 begin
  case which of
   scheduled: qlength:=qlength_s;
```

```
   poisson:  qlength:=qlength_p
  end;    {end case}
  i:=empty(queue,qlength);
  if i>0 then                    {i is at empty spot in queue, or}
   with queue[i] do               {zero if the queue is full}
    begin
     length:=packet_size;
     data_type:=which;
     delay:=-1
    end
  else
   lost:=lost+1;
 end;                          {end procedure}
{**********************************************************************
**********}
 begin                         {begin main program}
  rewrite(pro21a2);
  write(pro21a2,'These are the results of a simulation of a network in which');
  write(pro21a2,' terminal 1 acquires one long packet of scheduled data at');
  write(pro21a2,' the beginning of a cycle.  This long packet');
  write(pro21a2,' accounts for all the scheduled data.');
  write(pro21a2,'  The rest of the cycle is used by the network');
  write(pro21a2,' for transmission');
  write(pro21a2,' of poisson generated traffic according to an Ethernet scheme.');
  write(pro21a2,'  When terminal 1 receives scheduled data it must contend');
  writeln(pro21a2,' for the channel.');
  writeln(pro21a2);
  writeln(pro21a2,'N  =  number of stations  =',N);
  writeln(pro21a2,'packet length=',packet_length:0,' cycle length=',cycle_length:0);
  writeln(pro21a2,'# of transmissions that must be completed is',ST);
  fraction_scheduled:=0.1;
        long_packet:=trunc(fraction_scheduled/packet_length*cycle_length)*pack-
et_length+1;
   if realint(trunc(fraction_scheduled*cycle_length/packet_length))<fraction_sched-
uled*cycle_length/packet_length then
   long_packet:=long_packet+1;
  writeln(pro21a2,'fraction of scheduled traffic  =',fraction_scheduled:3:1);
  writeln(pro21a2,'This is program 2.');
  steps[1]:=0.1;
  steps[2]:=0.2;
  steps[3]:=0.3;
  steps[4]:=0.4;
  steps[5]:=0.5;
```

```
  steps[6]:=0.6;
  steps[7]:=0.7;
  steps[8]:=0.8;
  steps[9]:=0.9;
  steps[10]:=1.0;
  for z:=1 to 10 do
   begin
    rho:=steps[z];
    writeln(pro21a2);
    writeln(pro21a2,'rho =',rho:3:1);
    channel_busy:=false;
if rho>fraction_scheduled then
 begin
    transmissions:=0;   {init stats for each run}
    bkftime[scheduled]:=0;
    bkftime[poisson]:=0;
    bkoffs[scheduled]:=0;
    bkoffs[poisson]:=0;
    q_lost_s:=0;
    pkslst_p:=0;
    pkslst_s:=0;
    q_lost_p:=0;
    packets_generated:=0;
    etime:=0;
    cycles:=0;
    mtime:=0;
    mdelay:=0;
    mdelay_max:=0;
    mdelay_total:=0;
    scheduled_data:=false;
    lp_in_queue:=false;
  initialize(first,N,qlength_s,rho,SEEDi);
  while transmissions<ST do
   begin
    terminal:=first;
    ind:=0;
    transmitters:=0;
    busy:=channel_busy;
    while terminal<>nil do
     begin
      ind:=ind+1;
      with terminal^ do
       begin
```

```
      if backoff > 0 then backoff: = backoff-1;
      scheduled_data: = (mtime = 0) or scheduled_data;
      if scheduled_data and (empty(queue,qlength_s) > 0) and not lp_in_queue then
       begin
        lp_in_queue: = true;
      packets_generated: = packets_generated + round(realint(long_packet-1)/pack-
et_length);
        add_data(queue,long_packet,scheduled,q_lost_s)
       end
      else
       begin
         x: = uniform(0,1,SEEDi);
        if abs(x) < threshold then    {is there new data acquired during}
                                      {this slot time?}
         begin
          add_data(queue,packet_length,poisson,q_lost_p);
          packets_generated: = packets_generated + 1
         end
       end;
      i: = 1;
      while  (i < = qlength_s) and (queue[i].length > 0) do
      with queue[i] do
       begin
        if i = 1 then
         begin
          if transmitting then
           begin
            length: = length-1;
            if length = 0 then {has all of packet been transmitted?}
             begin
              busy: = false;
              transmitting: = false;
              transmissions: = transmissions + 1;
              for j: = 1 to qlength_s-1 do
               queue[j]: = queue[j + 1];
              queue[qlength_s].length: = 0;
             end;          {end if all of packet transmitted}
           end             {end if transmitting}
         end;              {end if i = 1}
         delay: = delay + 1;
         i: = i + 1;
       end;                {end with queue[i]}
      terminal: = next;
```

```
      end;                    {end with terminal}
    end;                        {end while terminal < > nil}
   channel_busy: = busy;
      if scheduled_data then
       mdelay: = mdelay + 1;
terminal: = first;
ind: = 0;
while terminal < > nil do
 begin
  with terminal^ do
           begin
            if (queue[1].length > 0) and  (not channel_busy) and (backoff = 0) then
                        {would this terminal like to transmit?}
             begin
              transmitters: = transmitters + 1;
              if transmitters > 1 then
               inbackoff(terminal^,SEEDi,pkslst_s,pkslst_p,bkoffs,bkftime);
             end;           {end if this terminal has data}
  terminal: = terminal^.next;
end;
           end;
   etime: = etime + 1;
   mtime: = mtime + 1;              {setting cycle counters}
      if mtime = (cycle_length-1) then
       begin
        mtime: = 0;
        cycles: = cycles + 1
       end;
   if (not channel_busy) and (transmitters > 0) then
    begin
     terminal: = first;
     while (terminal^.queue[1].length = 0) or (terminal^.backoff > 0) do
      terminal: = terminal^.next;
     with terminal^ do
      begin
       if transmitters = 1 then
        begin
         channel_busy: = true;
         transmitting: = true;
         attempts: = 0;
         backoff: = 0;
         begin_delay: = begin_delay + queue[1].delay;
         if queue[1].length > packet_length then
```

```
        begin
         if mdelay_max< mdelay then mdelay_max:= mdelay;
         mdelay_total:= mdelay_total+ mdelay;
         lp_in_queue:= false;
         mdelay:= 0;
         scheduled_data:= false
        end  {end if packet is scheduled data}
       end   {end if transmitting}
      else
       begin
        collisions:= collisions+ 1;
        inbackoff(terminal^,SEEDi,pkslst_s,pkslst_p,bkoffs,bkftime)
       end
     end;
   end;          {end with terminal^ }
 end;            {ST transmissions have now been simulated}
                   infotxms:= realint(transmissions)+ realint(cycles)*(fraction_sched-
uled*cycle_length/packet_length-1);
      transmissions:= transmissions+ round(realint(cycles)*((long_packet-1)/pack-
et_length));
      write(pro21a2,'Mean Delay = ',realint(begin_delay)/realint(transmis-
sions)*slot_time:7:5,' seconds');
 writeln(pro21a2,' = ',realint(begin_delay)/realint(transmissions):7:5,' slot times');
 write(pro21a2,'Collision Rate = ',realint(collisions)/realint(etime)/slot_time:7:5);
 writeln(pro21a2,' collisions/second');
  write(pro21a2,'Average poisson backoff time = ',realint(bkftime[poisson])/real-
int(bkoffs[poisson]):7:5);
   writeln(pro21a2,' slot times = ',realint(bkftime[poisson])/realint(bkoffs[pois-
son])*slot_time:7:5,' seconds');
    write(pro21a2,'Average scheduled backoff time = ',realint(bkftime[sched-
uled])/realint(bkoffs[scheduled]):7:5);
 writeln(pro21a2,' slot times = ',realint(bkftime[scheduled])/realint(bkoffs[sched-
uled])*slot_time:7:5,' seconds');
 writeln(pro21a2,etime:0,' slot times were simulated.');
    writeln(pro21a2,'Throughput = ',realint(transmissions)*packet_length/real-
int(etime)*100:7:5,'%');
    writeln(pro21a2,'Information throughput = ',infotxms*packet_length/real-
int(etime)*100:7:5,'%');
 write(pro21a2,'Maximum delay for scheduled data = ',mdelay_max:0,' slot times
= ');
 writeln(pro21a2,realint(mdelay_max)*slot_time:7:5,' seconds');
  write(pro21a2,'Average delay for scheduled data = ',realint(mdelay_total)/real-
int(cycles):7:5);
```

```
 writeln(pro21a2,' slot times = ',realint(mdelay_total)/realint(cycles)*slot_time:7:5,'
seconds');
 write(pro21a2,'Poisson generated packets lost due to insufficient queue');
 write(pro21a2,'space = ',q_lost_p:0,' = ');
 writeln(pro21a2,realint(q_lost_p)/realint(packets_generated)*100:0,'%');
 write(pro21a2,'Scheduled generated packets lost due to insufficient queue');
 write(pro21a2,'space = ',q_lost_s:0,' = ');
 writeln(pro21a2,realint(q_lost_s)/realint(packets_generated)*100:0,'%');
 writeln(pro21a2,cycles:0,' cycles were simulated');
 write(pro21a2,'Scheduled packets lost due to excessive delay = ');
 write(pro21a2,pkslst_s:0,' = ');
 write(pro21a2,realint(pkslst_s)/realint(cycles)*100:7:5,'%');
 writeln(pro21a2,' of scheduled packets');
 write(pro21a2,'Poisson packets lost due to excessive delay = ');
 write(pro21a2,pkslst_p:0,' = ');
 write(pro21a2,realint(pkslst_p)/realint(transmissions)*100:7:5,'%');
 writeln(pro21a2,' of all packets (poisson and scheduled)');
end;
end;
 close(pro21a2);
end.
```

B.4 MFA SIMULATOR PROGRAM THREE

This program implements the MFA protocol and allows scheduled data to gain control of the channel as soon as the bus is clear after the beginning of a cycle.

```
program MNet_sim(input,output);
{This program simulates and Ethernet network where for there is scheduled data}
{based on a cycle of length cycle_length. When a terminal acquires scheduled }
{data, it will gain access to the channel as soon as the transmission in prog-}
{ress (when the terminal acquires the scheduled data) is completed.}
 const
  gap=10;              {distance, in meters, between terminals}
  packet_length=1;     {length of packet in slot-times}
  slot_time=51.2e-6;   {round-trip propagation delay, in seconds}
  N=40;                {number of terminals}
  qlength_p=1;     {number of poisson generated packets that can be}
                   {stored by a terminal}
  qlength_s=5;     {number of scheduled packets that can be}
```

```
                {stored by a terminal}
 ST=50000;          {number of successful transmissions that must be}
                {completed for the simulation}
 pi=3.141592654;
 cycle_length=100;   {length of MFA cycle in slot-times}
{**********************************************************************
**********}
 type
  how_generated=(scheduled,poisson);
  bkfcount=array[how_generated] of integer;
  ptr=^node;
  packet=record
   length:integer;          {number of slot times required to transmit re- }
                        {mainder of the packet.}
   delay:integer;           {number of slot times between generation of }
                        {packet and transmission}
   data_type:how_generated    {indicates whether the packet is scheduled,}
                        {or poisson-generated data}
  end;
  data=array[1..qlength_s] of packet;
  node=record               {Each node in linked lists contains information}
                        {describing one terminal on the network.}
   transmitting:boolean;      {transmitting flag (is terminal transmitting)}
   queue:data;              {info. describing queued pack-}
                        {ets}
   backoff:integer;         {slot-times until terminal may again attempt}
                        {transmission}
   attempts:integer;        {number of attempts at transmission of packet}
                        {described by queue[1] }
   distance:integer;        {distance to previous terminal}
   threshold:real;          {used to determine if terminal has data arri-}
                        {ving during a slot time}
   next:ptr;              {pointer to next node on network}
  end;
{**********************************************************************
**********}
 var
  begin_delay:integer;        {sum of all delays experienced by transmitted}
                        {packets}
  collisions:integer;         {number of collisions}
  rho:real;                 {normalized throughput of the network}
  steps:array[1..12] of real;  {values of rho}
  channel_busy,busy:boolean;   {is the channel being used?}
```

```
 etime:integer;              {number of slot times simulated}
 mtime:integer;              {mfa cycle time counter}
 transmissions:integer;      {number of successful transmissions completed}
 transmitters: integer;      {number of terminals with data to transmit in a}
                        {given slot time}
 terminal:ptr;               {pointer to a terminal}
 first:ptr;                  {pointer to the first terminal}
 pkslst_s,pkslst_p:integer;  {total number of scheduled and poisson}
                        {packets lost due to excessive delay}
 q_lost_p,q_lost_s:integer;        {total number of packets lost due to insufficient-}
                        {queue space}
 packets_generated:integer;  {total number of packets generated}
 SEEDi:integer;              {seeds for random number generators}
 i,j,ind,z:integer;
 x:real;
 fraction_scheduled:real;    {ratio of scheduled traffic to bandwidth}
 scheduled_data:boolean;     {flag indicating the beginning of a cycle}
 sched_at_head:boolean;  {flag indicating that scheduled data is at the head of the
line}
 lp_in_queue:boolean;    {flag indicating scheduled data has been placed in a queue}
 mdelay_max,mdelay,mdelay_total:integer;  {maximum, delay during cycle and }
                        {total delay experienced by scheduled data}
                        {(in slot times)}
 tx_length:integer;          {used in determining if a terminal may transmit}
 cycles:integer;             {number of cycles simulated}
 long_packet:integer;        {the length of the scheduled, extended packet}
 infotxms:real;  {number of packet of useful information transmitted}
 bkoffs,bkftime:bkfcount;  {number of times terminals go into backoff, }
                      {and total amount of time spent in backoff}
{**********************************************************************
**********}
 function realint(n:integer):real;
  begin
   realint:=n;
  end;
{**********************************************************************
**********}
 procedure initialize(var first:ptr;N,qlength:integer;rho:real;
                                              var SEEDi:integer);
  var
   i,j:integer;
   terminal:ptr;
   t:real;
```

```
begin
 SEEDi:=56912;
 t:=(rho-fraction_scheduled)/N/packet_length;
 new(first);
 terminal:=first;
 for i:=1 to N do
  begin
   with terminal^ do
    begin
     transmitting:=false;
     distance:=gap;
     attempts:=0;
     backoff:=0;
     threshhold:=t;
     for j:=1 to qlength do
      with queue[j] do
       begin
        length:=0;
        delay:=0
       end
    end;
   if (i=N) then
    terminal^.next:=nil
   else
    begin
     new(terminal^.next);
     terminal:=terminal^.next;
    end
  end
 end;
{**********************************************************************
**********}
 procedure randint(var SEED,n:integer;limit:integer);
{This procedure produces a random integer in the variable n where 0<n<limit.}
 begin
  n:=1+round(realint(limit-1)*random(SEED));
 end;
{**********************************************************************
**********}
 procedure inbackoff(var terminal:node; var SEED,pkslst_s,pkslst_p:integer; var
bkoffs,bkftime:bkfcount);
 var
  i:integer;
```

```
begin
 with terminal do
  begin
   attempts:=attempts+1;
   if attempts>16 then
    begin
     attempts:=0;
     case queue[1].data_type of
      poisson: pkslst_p:=pkslst_p+1;
      scheduled: pkslst_s:=pkslst_s+1;
     end;  {end case}
     for i:=1 to (qlength_s-1) do
      queue[i]:=queue[i+1];
     with queue[qlength_s] do
      begin
       length:=0;
       delay:=0
      end
    end
   else
    begin
    bkoffs[queue[1].data_type]:=bkoffs[queue[1].data_type]+1;
    if attempts<=8 then
     randint(SEED,backoff,round(exp(attempts*ln(2))))
    else
     randint(SEED,backoff,256);
    bkftime[queue[1].data_type]:=bkftime[queue[1].data_type]+backoff
    end
  end
 end;
{**************************************************************************
**********}
 function empty(queue:data;qlength:integer):integer;
{This function checks to see if there is an empty spot on the queue, and if}
{so, returns the place of that spot.  If there is no empty spot, the function}
{returns a zero.}
 var
  i:integer;
 begin
  i:=1;
  if qlength>1 then
    while  (i<qlength) and(queue[i].length>0) do
     i:=i+1;
```

```
  if queue[i].length=0 then
   empty:=i
  else
   empty:=0;
 end;
{**********************************************************************
**********}
 procedure add_data(var queue:data;packet_size:integer;which:how_generated;var
lost:integer);
 var
 i,qlength:integer;
 begin
 case which of
  scheduled: qlength:=qlength_s;
  poisson:  qlength:=qlength_p
 end:   {end case}
  i:=empty(queue,qlength);
  if i>0 then                {i is at empty spot in queue, or}
   with queue[i] do           {zero if the queue is full}
    begin
     length:=packet_size;
     data_type:=which;
     delay:=-1
    end
  else
   lost:=lost+1;
 end:                       {end procedure}
{**********************************************************************
**********}
 begin                      {begin main program}
  write('These are the results of a simulation of a network in which');
  write(' a terminal acquires one long packet of scheduled data at');
  write(' the beginning of a cycle.  This long packet');
  write(' accounts for all the scheduled data.');
  write('  The rest of the cycle is used by the network');
  write(' for transmission');
  write(' of poisson generated traffic according to an Ethernet scheme.');
  write('  When a terminal receives scheduled data it will have acc');
  write('ess to the channel as soon as it is free i.e. the termi');
  writeln('nal with the scheduled data does not have to contend.');
  writeln;
  writeln('N = number of stations = ',N);
  writeln('packet length= ',packet_length:0,'  cycle length= ',cycle_length:0);
```

```
  writeln('# of transmissions that must be completed is',ST);
  fraction_scheduled:=0.1;
        long_packet:=trunc(fraction_scheduled/packet_length*cycle_length)*pack-
et_length+1;
   if realint(trunc(fraction_scheduled*cycle_length/packet_length))<fraction_sched-
uled*cycle_length/packet_length then
    long_packet:=long_packet+1;
  writeln('fraction of scheduled traffic =',fraction_scheduled:3:1);
  writeln('This is program 3.');
  steps[1]:=0.1;
  steps[2]:=0.2;
  steps[3]:=0.3;
  steps[4]:=0.4;
  steps[5]:=0.5;
  steps[6]:=0.6;
  steps[7]:=0.7;
  steps[8]:=0.8;
  steps[9]:=0.9;
  steps[10]:=1.0;
  for z:=1 to 10 do
   begin
    rho:=steps[z];
    writeln;
    writeln('rho =',rho:3:1);
    channel_busy:=false;
if rho>fraction_scheduled then
 begin
    transmissions:=0;   {init stats for each run}
    bkftime[scheduled]:=0;
    bkftime[poisson]:=0;
    bkoffs[scheduled]:=0;
    bkoffs[poisson]:=0;
    q_lost_s:=0;
    pkslst_p:=0;
    pkslst_s:=0;
    q_lost_p:=0;
    packets_generated:=0;
    etime:=0;
    cycles:=0;
    mtime:=0;
    mdelay:=-1;
    mdelay_max:=0;
    mdelay_total:=0;
```

```
initialize(first,N,qlength_s,rho,SEEDi);
while transmissions < ST do
 begin
  terminal: = first;
  ind: = 0;
  transmitters: = 0;
  busy: = channel_busy;
  scheduled_data: = false;
  sched_at_head: = false;
  lp_in_queue: = false;
  while terminal < > nil do
   begin
    ind: = ind + 1;
    with terminal^ do
     begin
      if backoff > 0 then backoff: = backoff-1;
      scheduled_data: = (mtime = 0) or scheduled_data;
      if scheduled_data and (empty(queue,qlength_s) > 0) and not lp_in_queue then
       begin
        lp_in_queue: = true;
      packets_generated: = packets_generated + round(realint(long_packet-1)/pack-
et_length);
        sched_at_head: = empty(queue,qlength_s) = 1;
      if sched_at_head then
       begin
        if (not channel_busy) then transmitters: = 1
       end;
        add_data(queue,long_packet,scheduled,q_lost_s);
       end
      else
        begin
          x: = random(SEEDi);
         if abs(x) < threshold then    {is there new data acquired during}
                                       {this slot time?}
          begin
           add_data(queue,packet_length,poisson,q_lost_p);
           packets_generated: = packets_generated + 1
          end
        end;
      i: = 1;
      while  (i < = qlength_s) and (queue[i].length > 0) do
      with queue[i] do
        begin
```

```
          if i=1 then
           begin
            if transmitting then
             begin
              length:=length-1;
              if length=0 then {has all of packet been transmitted?}
               begin
                busy:=false;
                transmitting:=false;
                transmissions:=transmissions+1;
                for j:=1 to qlength_s-1 do
                 queue[j]:=queue[j+1];
                queue[qlength_s].length:=0;
               end;          {end if all of packet transmitted}
             end             {end if transmitting}
           end;              {end if i=1}
           delay:=delay+1;
           i:=i+1;
         end;                {end with queue[i]}
       terminal:=next;
      end;                   {end with terminal}
    end;                        {end while terminal<>nil}
   channel_busy:=busy;
       if scheduled_data then mdelay:=mdelay+1;
terminal:=first;
ind:=0;
while terminal<>nil do
 begin
  with terminal^ do
          begin
          if (queue[1].length>0) and  (not channel_busy) and (backoff=0) and (not
sched_at_head) then
                        {would this terminal like to transmit?}
            begin
             transmitters:=transmitters+1;
             if transmitters>1 then
              inbackoff(terminal^,SEEDi,pkslst_s,pkslst_p,bkoffs,bkftime);
            end;          {end if this terminal has data}
  terminal:=terminal^.next;
end;
          end;
   etime:=etime+1;
   mtime:=mtime+1;            {setting cycle counters}
```

```
      if mtime = (cycle_length-1) then
       begin
        if scheduled_data then
writeln('!!!!!!Scheduled data overlap ERROR!!!!!!!');
        mtime: = 0;
        cycles: = cycles + 1
       end;
  if (not channel_busy) and (transmitters > 0) then
   begin
    terminal: = first;
    if sched_at_head then
     begin
     sched_at_head: = false;
     tx_length: = packet_length + 1
     end
    else
     tx_length: = 1;
    while (terminal^.queue[1].length < tx_length) or (terminal^.backoff > 0) do
     begin
     terminal: = terminal^.next;
     if (terminal = nil) and (tx_length < > 1) then
      begin
     terminal: = first;
     tx_length: = 1;
    end
     end;
    with terminal^ do
     begin
      if transmitters = 1 then
       begin
        channel_busy: = true;
        transmitting: = true;
        attempts: = 0;
        backoff: = 0;
        begin_delay: = begin_delay + queue[1].delay;
        if queue[1].length > packet_length then
         begin
          if mdelay_max < mdelay then mdelay_max: = mdelay;
          mdelay_total: = mdelay_total + mdelay;
          lp_in_queue: = false;
          mdelay: = -1;
          scheduled_data: = false
         end  {end if packet is scheduled data}
```

```
      end   {end if transmitting}
     else
      begin
       collisions:=collisions+1;
       inbackoff(terminal^,SEEDi,pkslst_s,pkslst_p,bkoffs,bkftime)
      end
    end;
  end;          {end with terminal^}
 end;           {ST transmissions have now been simulated}
                 infotxms:=realint(transmissions)+realint(cycles)*(fraction_sched-
uled*cycle_length/packet_length-1);
      transmissions:=transmissions+round(realint(cycles)*((long_packet-1)/pack-
et_length));
 write('Mean Delay = ',realint(begin_delay)/realint(transmissions)*slot_time:7:5,'
seconds');
 writeln(' = ',realint(begin_delay)/realint(transmissions):7:5,' slot times');
 write('Collision Rate = ',realint(collisions)/realint(etime)/slot_time:7:5);
 writeln(' collisions/second');
   write('Average poisson backoff time = ',realint(bkftime[poisson])/real-
int(bkoffs[poisson]):7:5);
    writeln(' slot times = ',realint(bkftime[poisson])/realint(bkoffs[pois-
son])*slot_time:7:5,' seconds');
  write('Average scheduled backoff time = ',realint(bkftime[scheduled])/real-
int(bkoffs[scheduled]):7:5);
   writeln(' slot times = ',realint(bkftime[scheduled])/realint(bkoffs[sched-
uled])*slot_time:7:5,' seconds');
 writeln(etime:0,' slot times were simulated.');
       writeln('Throughput = ',realint(transmissions)*packet_length/real-
int(etime)*100:7:5,'%');
      writeln('Information throughput = ',infotxms*packet_length/real-
int(etime)*100:7:5,'%');
 write('Maximum delay for scheduled data = ',mdelay_max:0,' slot times = ');
 writeln(realint(mdelay_max)*slot_time:7:5,' seconds');
   write('Average delay for scheduled data = ',realint(mdelay_total)/real-
int(cycles):7:5);
 writeln(' slot times = ',realint(mdelay_total)/realint(cycles)*slot_time:7:5,' sec-
onds');
 write('Poisson generated packets lost due to insufficient queue');
 write('space = ',q_lost_p:0,' = ');
 writeln(realint(q_lost_p)/realint(packets_generated)*100:0,'%');
 write('Scheduled generated packets lost due to insufficient queue');
 write('space = ',q_lost_s:0,' = ');
 writeln(realint(q_lost_s)/realint(packets_generated)*100:0,'%');
```

```
  writeln(cycles:0,' cycles were simulated');
  write('Scheduled packets lost due to excessive delay = ');
  write(pkslst_s:0,' = ');
  write(realint(pkslst_s)/realint(cycles)*100:7:5,'%');
  writeln(' of scheduled packets');
  write('Poisson packets lost due to excessive delay = ');
  write(pkslst_p:0,' = ');
  write(realint(pkslst_p)/realint(transmissions)*100:7:5,'%');
  writeln(' of all packets (poisson and scheduled)');
 end;
end;
end.
```

APPENDIX C MATH MODEL CALCULATIONS

C.1 MATH MODEL NOTES

A program was written to solve the equations given for throughput and delay of the CSMA/CD protocol in Chapter 5 for different values of the parameters σ and ν. The program is presented in section C.2 and the output of several trials are presented in section C.3. The program was written for the modified Sik as defined in Chapter 5 of the text. The Sik used for the calculation of the average backlog during tranmission ,A(i), is as defined by Tobagi.

C.2 MATH MODEL PROGRAM

```
      double precision qt(0:100,0:100),xq(0:100,0:100)
      double precision sqt(0:100,0:100),xqq(0:100,0:100)
      double precision ps(0:100),xpp(0:100,0:100),tr1(0:100,0:100)
      double precision ter2(0:100,0:100),sumt(0:100,0:100)
      double precision pi(0:100),sumg(0:100,0:100)
      double precision xmu(0:100),xqu(0:100),a(0:100)
      double precision xqc(0:100),xmr(0:100),xss(0:100,0:100)
      double precision xff(0:100,0:100),xqg(0:100,0:100)
      double precision xm1(0:100,0:100),xm2(0:100,0:100)
      double precision dnd(0:100,0:100),xsd(0:100,0:100)
      common xqc,xmr,l,xss
      common xff,xqg,idd,sum,iq,iv
      common igap,xm1,xm2
c     print*,'input m'
      read*,m
      do 501 i=0,m
      do 601 k=0,m
      sumt(i,k)=0.0
      sumg(i,k)=0.0
  601 continue
  501 continue
c     print*,'input sigma'
      read*,sig
c     print*,'input packet length in slots'
```

```
      read*,id
      print*,'the average traffic on the bus is',sig
      sig=sig/m
c     print*,'the number of stations =',m
      sig=sig/id
      print*,'probabilty station will generate traffic in slot',sig
      l=m-1
c********** generating the q matrix xq **************
c     print*,'generating q matrix'
      do 22 ii=0,m
      do 33 ki=0,m
      if(ki.lt.ii)then
      xq(ii,ki)=0.0
      xqq(ii,ki)=0.0
      go to 44
      else
      rmm=m-ii
      rkk=ki-ii
      call bino(rmm,rkk,sig,r)
      xq(ii,ki)=r
      xqq(ii,ki)=r
      xm2(ii,ki)=r
 44   continue
      end if
 33   continue
 22   continue
c     print*,'**** generating q**t+1 matrix xqq and q**gama+1 xqg ****
c     print*,'generating q**t+1 and q**gama+1'
c     print*,'input gama ?'
      read*,iga
      igap=iga+1
      do 80 idd=1,id
      do 31 iv=0,m
      ia=0
      do 13 i=0,m
      xmr(ia)=xqq(iv,i)
c     print*,xmr(0,ia)
      ia=1+ia
 13   continue
      do 15 iq=0,m
      ib=0
      do 14 j=0,m
      xqc(ib)=xq(j,iq)
```

```
      ib=ib+1
   14 continue
      call matm(m,xmr,sum,xqc,iv,iq,idd,igap)
      qt(iv,iq)=sum
      xqq(iv,iq)=sum
c     print*,'xqq(',iv,iq,')=',xqq(iv,iq)
       if(idd.ge.igap)go to 15
      xqg(iv,iq)=sum
c     print*,'xqg(',iv,iq,')=',xqg(iv,iq)
   15 continue
   31 continue
      call matad(m,sumt,xqq,sumt)
      if(idd.gt.igap)go to 80
      call matad(m,sumg,xqg,sumg)
c     print*,'sumt(',i,k,')=',sumt(i,k)
   80 continue
      do 909 iva=0,m
c     print*,'qt+1(',iva,'0)=',xqq(iva,0)
c     print*,'qg+1(',iva,'0)=',xqg(iva,0)
  909 continue
c     **** generating j matrix xjj ****
c     **** generating s matrix xss ****
c     print*,'generating s matrix'
c     print*,'input nu'
      read*,xn
      xnd=1.-xn
      sigd=1.-sig
      do 34 js=0,m
      do 23 is=0,m
      mi=m-is
      iss=is-1
      mii=mi-1
      iis=is+1
      if(js.eq.iss)then
      xss(is,js)=((sigd**mi)*(is*xn*xnd**iss))/(1-(xnd**is)*(sigd**mi))
      else
      if(js.eq.is) then
      xss(is,js)=((mi*sig*sigd**mii)*xnd**is)/(1-(xnd**is)*(sigd**mi))
      else
      xss(is,js)=0.
      end if
      end if
   23 continue
```

```
 34   continue
      do 808 ixa=0,m
c     print*,'s(0',ixa,')=',xss(0,ixa)
 808  continue
c     **** generating sd matrix xsd ****
c     print*,'generating sd matrix'
      do 834 js=0,m
      do 823 is=0,m
      mi=m-is
      iss=is-1
      mii=mi-1
      iis=is+1
      if(js.eq.is)then
      xsd(is,js)=(sigd**mi)*(is*xn*xnd**iss)/(1-(xnd**is)*(sigd**mi))
      else
      if(js.eq.iis) then
      xsd(is,js)=((mi*sig*sigd**mii)*xnd**is)/(1-(xnd**is)*(sigd**mi))
      else
      xsd(is,js)=0.
      end if
      end if
 823  continue
 834  continue
c     **** generating f matrix xff ****
c     print*,'generating f matrix'
      do 55 jf=0,m
      do 65 iq=0,m
      mf=m-iq
      iff=iq-1
      mmf=mf-1
      mff=m-jf
      ji=jf-iq
      igg=iq+1
      if(jf.eq.iq)then
            xff(iq,jf)=(sigd**mf)
            tempf=xnd**iff
            tempf=iq*xn*tempf
            tempf=tempf+xnd**iq
            tempf=1.0-tempf
            xff(iq,jf)=xff(iq,jf)*tempf
      xff(iq,jf)=xff(iq,jf)/(1.0-(xnd**iq)*(sigd**mf))
      else
      if(jf.gt.igg) then
```

```
      rmm = m-iq
      rkk = jf-iq
      call bino(rmm,rkk,sig,r)
      xff(iq,jf) = r/(1-(xnd**iq)*(sigd**mf))
      else
      if(jf.eq.igg) then
      xff(iq,jf) = (mf*sig*(sigd**mmf)*(1-xnd**iq))
      xff(iq,jf) = xff(iq,jf)/(1-(xnd**iq)*(sigd**mf))
      else
      xff(iq,jf) = 0.
      end if
      end if
      end if
   65 continue
   55 continue
      total = 0.
      do 2502 ip = 0,m
      total = 0.0
      do 2602 jp = 0,m
      total = total + xff(ip,jp)
 2602 continue
c     print*,'xff(',ip,'total) = ',total
 2502 continue
c     **** (s)*(q**t+1) *****
c     print*,'generating s*q**t+1'
      do 310 iv = 0,m
      ia = 0
      do 130 i = 0,m
      xmr(ia) = xss(iv,i)
      ia = 1 + ia
  130 continue
      do 150 iq = 0,m
      ib = 0
      do 140 j = 0,m
      xqc(ib) = xqq(j,iq)
      ib = ib + 1
  140 continue
      call matm(m,xmr,sum,xqc,iv,iq,idd,igap)
      sqt(iv,iq) = sum
      tr1(iv,iq) = sqt(iv,iq)
  150 continue
  310 continue
      total = 0.
```

```
      do 1502 ip=0,m
      total=0.
      do 1602 jp=0,m
      total=total+sqt(ip,jp)
 1602 continue
c     print*,'sqt(',ip,'total)=',total
 1502 continue
c     **** (xff)*(q**gama+1)=ter2 ****
c     print*,'generating xff*q**gama+1'
      do 710 iv=0,m
      ia=0
      do 730 i=0,m
      xmr(ia)=xff(iv,i)
      ia=ia+1
  730 continue
      do 750 iq=0,m
      ib=0
      do 740 j=0,m
      xqc(ib)=xqg(j,iq)
      ib=ib+1
  740 continue
      call matm(m,xmr,sum,xqc,iv,iq,idd,igap)
      ter2(iv,iq)=sum
c     print*,'ter2(',iv,iq,')=',ter2(iv,iq)
750   continue
710   continue
c     print*,'ter2=',ter2(0,0)
c     **** tr1+ter2=xpp ****
c     print*,'generating xpp'
      do 800 ix=0,m
      do 801 jx=0,m
      xm1(ix,jx)=tr1(ix,jx)
      xm2(ix,jx)=ter2(ix,jx)
  801 continue
  800 continue
      call matad(m,xm1,xm2,xpp)
      total=0.
      do 502 ip=0,m
      total=0.
      do 602 jp=0,m
      total=total+xpp(ip,jp)
      if (jp.ge.ip-1.and.jp.le.ip+1) then
      print*,'xpp(',ip,jp,')=',xpp(ip,jp)
```

```
      goto 1234
      else
 1234 continue
      end if
  602 continue
c     print*,'xpp(',ip,'total)=',total
  502 continue
c     **** ps calculation ****
      print*,'generating ps'
      do 905 ip=0,m
      mi=m-ip
c     print*,'mi(',ip,') =',mi
      mi1=m-ip-1
c     print*,'mi1 =',mi1
      i1=ip-1
c     print*,'i1 =',i1
      ps(ip)=(mi*sig)*(sigd**mi1)*(xnd**ip)+(ip*xn)*(xnd**i1)*(sigd**mi)
c     print*,'psp(',ip,')=',ps(ip)
      aterm=(xnd**ip)
c     print*,'aterm =',aterm
      bterm=(sigd**mi)
c     print*,'bterm =',bterm
      cterm=aterm*bterm
c     print*,'cterm =',cterm
      xpn=1.0-cterm
c     print*,'xpn(',ip,') =',xpn
      ps(ip)=ps(ip)/xpn
      print*,'ps(',ip,')=',ps(ip)
  905 continue
c     **** pi calculations *****
c     print*,'generating pi'
      pi(0)=1.0
      pi(1)=pi(0)-xpp(0,0)
      pi(1)=pi(1)/xpp(1,0)
      print*,'pi(1)=',pi(1)
      do 123 j=1,m-1
      jj=j+1
      i=0
   99 sump=0.0
      if(i.gt.j)go to 322
      do 321 ii=i,j
      sump=sump+pi(ii)*xpp(ii,j)
  321 continue
```

```
  322 pi(jj)=pi(j)-sump
      pi(jj)=pi(jj)/xpp(jj,j)
c 127 print*,'pi(',jj,')=',pi(jj)
  123 continue
      print*,'generate and normalize the sum of pi to 1'
      sumn=0.0
      do 456 i=0,m
      sumn=pi(i)+sumn
  456 continue
      do 567 i=0,m
      pi(i)=pi(i)/sumn
      print*,'pi(',i,')=',pi(i)
  567 continue
      sumn=0.0
      do 1456 i=0,m
      sumn=pi(i)+sumn
 1456 continue
      print*,'pi(total)=',sumn
c     ***** numerator of throughput equation ****
      snum=0.0
      do 78 i=0,m
      snum=snum+pi(i)*ps(i)
   78 continue
      snum=snum*id
c     **** denomenator of throughput equation ****
      sden=0.0
      do 87 i=0,m
      mi=m-i
      del=xnd**i*sigd**mi
      deli=1./(1.-del)
c     print*,'deli =',deli
      sden=sden+pi(i)*(deli+1.+ps(i)*(id-iga)+iga)
   87 continue
c     ***** throughput ****
            s=snum/sden
            print*,'throughput=',s,'nu=',xn,'packet=',id,'st=',m
            print*,'snum =',snum
            print*,'sden =',sden
c     **** Delay ****
      do 904 j=0,m
      do 906 k=0,m
      if(j.eq.k)then
      dnd(j,k)=1.0
```

```
      else
      dnd(j,k)=0.0
      end if
906   continue
904   continue
      call matad(m,dnd,xq,dnd)
      call matad(m,dnd,sumt,sumt)
      call matad(m,dnd,sumg,sumg)
      do 71 iv=0,m
      ia=0
      do 73 i=0,m
      xmr(ia)=xsd(iv,i)
      xmu(ia)=xff(iv,i)
      ia=ia+1
73    continue
      do 75 iq=0,m
      ib=0
      do 74 j=0,m
      xqc(ib)=sumt(j,iq)
      xqu(ib)=sumg(j,iq)
      ib=ib+1
74    continue
      call matm(m,xmr,sum,xqc,iv,iq,idd)
      sumt(iv,iq)=sum
      call matm(m,xmu,sum,xqu,iv,iq,idd)
      sumg(iv,iq)=sum
75    continue
71    continue
      call matad(m,sumt,sumg,sumt)
      suma=0.
      tempa=0.
      do 108 i=0,m
      mi=m-i
      suma=0.
      do 208 j=i,m
      print*,'sumt(',i,j,')=',sumt(i,j)
      suma=suma+real(j)*sumt(i,j)
208   continue
      a(i)=suma
      print*,'A(',i,')=',a(i)
      del=xnd**i*sigd**mi
      print*,'i =',i
      print*,'reali =',real(i)
```

```
      deli = real(i)/(1.-del)
      xna = deli + a(i)
      print*,'xna  = ',xna
      tempa = tempa + xna*pi(i)
      print*,'tempa(',i,') = ',tempa
  108 continue
      delay = tempa/snum
      print*,'delay = ',delay,'sigma = ',sig
      print*,' '
      stop
      end

      subroutine bino(rmm,rkk,sig,r)
      ridl = rmm-rkk
      sigd = 1.-sig
      r = (sig**rkk)*(sigd**ridl)
      rn = rmm
      call fact(rn,rnr)
      rnum = rnr
      rn = rkk
      call fact(rn,rnr)
      rndn = rnr
      ridl = rmm-rkk
      rn = ridl
      call fact(rn,rnr)
      rndn = rndn*rnr
      r = r*rnum/rndn
      return
      end
      subroutine fact(rn,rnr)
      rnr = 1.
      do 11 ri = 0,rn-2.
      rnr = rnr*(rn-ri)
c     print*,'rnr = ',rnr
      11   continue
      return
      end
      subroutine matm(m,xmr,sum,xqc,iv,iq,idd,igap)
      double precision xmr(0:100),xqc(0:100)
      sum = 0.0
      do 99 ip = 0,m
      sum = sum + xmr(ip)*xqc(ip)
      99     continue
```

```
      return
      end
      subroutine matad(m,xm1,xm2,xr)
      double precision xm1(0:100,0:100),xm2(0:100,0:100)
      double precision xr(0:100,0:100)
      do 37 ia=0,m
      do 38 ja=0,m
      xr(ia,ja)=xm1(ia,ja)+xm2(ia,ja)
 38   continue
 37   continue
      return
      end
```

C.3 Math Model Program Outputs

The output of several trials of the math model program are presented in Tables C.1 and C.2. The values for $X_{pp}(i,k)$ indicate the probability that given there are i stations in backlog at the beginning of a transmission there will be k stations in backlog at the completion of the transmission. The value $\Pi(i)$ indicates the probability that i stations will be in backlog. The values of other intermediate probabilities, such as the probability of success given i stations in backlog, $P_s(i)$, are not tabulated but several of each type were calculated by hand to debug the program. The values of ν are fixed and this varies from the Ethernet and MFA protocols where the backoff interval is set by the station and changes as system load changes.

The values for xpp(i,k), and $\Pi(i)$ follow the correct trends and except for $\Pi(i)$ for large values of i where the value is negative. The value for throughput, $\overline{S}$, is a good approximation. The values for delay, $\overline{D}$, should be larger than one since they are normalized to the packet transmission time. The values for throughput are correct and the average number of stations in backlog while the channel is idle is correct. For large packet sizes, ps, the delays are above one. The model gives poor performance when the average transmission time is less than 10 slot times.

Table C.1 State Transition Probabilities

σ	0.1000	0.1000	0.1000	0.3000	0.5000	0.1000	0.1000
ν	0.1000	0.1500	0.0500	0.1000	0.1000	0.1000	0.1000
ps	16	16	16	16	16	10	48
i,k	xpp(i,k)						
0,0	0.8965	0.8965	0.8965	0.7205	0.5790	0.8915	0.9019
1,0	0.8486	0.8648	0.8035	0.6162	0.4515	0.8177	0.8854
1,1	0.1360	0.1214	0.1768	0.2985	0.3673	0.1606	0.1065
2,1	0.8317	0.8141	0.8326	0.6422	0.4970	0.8146	0.8514
2,2	0.1556	0.1736	0.1529	0.2916	0.3612	0.1675	0.1418
3,2	0.7992	0.7553	0.8315	0.6353	0.5054	0.7873	0.8126
3,3	0.1892	0.2329	0.1561	0.3077	0.3733	0.1960	0.1813
4,3	0.7636	0.6970	0.8224	0.6194	0.5026	0.7546	0.7737
4,4	0.2255	0.2915	0.1666	0.3298	0.3907	0.2295	0.2208
10,9	0.5584	0.4064	0.7350	0.4920	0.4335	0.5556	0.5614
10,10	0.4338	0.5845	0.2585	0.4795	0.5122	0.4323	0.4354
11,10	0.5279	0.3687	0.7195	0.4708	0.4198	0.5256	0.5304
11,11	0.4649	0.6229	0.2746	0.5037	0.5323	0.4632	0.4668
12,11	0.4985	0.3338	0.7040	0.4499	0.4060	0.4967	0.5006
12,12	0.4949	0.6585	0.2908	0.5275	0.5522	0.4931	0.4968
13,12	0.4704	0.3016	0.6886	0.4295	0.3922	0.4689	0.4721
13,13	0.5237	0.6915	0.3068	0.5507	0.5717	0.5219	0.5257
14,13	0.4435	0.2721	0.6733	0.4097	0.3784	0.4423	0.4449
14,14	0.5513	0.7219	0.3227	0.5733	0.5910	0.5496	0.5533
15,14	0.4177	0.2450	0.6582	0.3904	0.3648	0.4168	0.4188
15,15	0.5779	0.7498	0.3385	0.5954	0.6100	0.5763	0.5796
19,18	0.3261	0.1585	0.5993	0.3190	0.3121	0.3259	0.3263
19,19	0.6729	0.8404	0.4000	0.6781	0.6830	0.6726	0.6733
20,19	0.3059	0.1416	0.5851	0.3027	0.2995	0.3059	0.3060
20,20	0.6941	0.8584	0.4149	0.6973	0.7005	0.6941	0.6940
$\overline{S}$	0.0999	0.0999	0.0998	0.2976	0.4842	0.1000	0.1003
$\overline{D}$	1.2809	1.2576	1.3581	1.7199	2.4911	1.4398	1.1729

Table C.2 Performance Characteristics of the Network

σ	0.1000	0.1000	0.1000	0.3000	0.5000	0.1000	0.1000
ν	0.1000	0.1500	0.0500	0.1000	0.1000	0.1000	0.1000
ps	16	16	16	16	16	10	48
i	$\Pi(i)$						
0	0.8827	0.8851	0.8764	0.6161	0.3465	0.8682	0.8870
1	0.1076	0.1059	0.1128	0.2795	0.3231	0.1152	0.0983
2	0.0104	0.0104	0.0110	0.0831	0.1939	0.0139	0.0070
3	0.0004	0.0004	0.0005	0.0171	0.0872	0.0009	0.0005
4	–0.0000	–0.0000	–0.0000	0.0030	0.0336	0.0001	0.0003
5	–0.0000	–0.0000	–0.0000	0.0005	0.0115	0.0001	0.0003
6	–0.0000	–0.0001	–0.0000	0.0001	0.0035	0.0001	0.0003
7	–0.0000	–0.0001	–0.0000	0.0000	0.0010	0.0001	0.0003
8	–0.0001	–0.0001	–0.0000	0.0000	0.0002	0.0001	0.0003
9	–0.0001	–0.0001	–0.0000	0.0000	0.0000	0.0001	0.0003
10	–0.0001	–0.0001	–0.0000	0.0000	–0.0000	0.0001	0.0004
11	–0.0001	–0.0001	–0.0000	0.0000	–0.0000	0.0001	0.0004
12	–0.0001	–0.0001	–0.0000	0.0000	–0.0000	0.0001	0.0004
13	–0.0001	–0.0001	–0.0000	0.0000	–0.0000	0.0001	0.0004
14	–0.0001	–0.0001	–0.0000	0.0000	–0.0000	0.0001	0.0005
15	–0.0001	–0.0001	–0.0000	0.0000	–0.0001	0.0001	0.0005
16	–0.0001	–0.0001	–0.0000	0.0000	–0.0001	0.0001	0.0005
17	–0.0001	–0.0002	–0.0000	0.0000	–0.0001	0.0001	0.0005
18	–0.0001	–0.0002	–0.0001	0.0000	–0.0001	0.0001	0.0006
19	–0.0001	–0.0002	–0.0001	0.0001	–0.0001	0.0002	0.0006
20	–0.0001	–0.0002	–0.0001	0.0001	–0.0001	0.0002	0.0007
$\overline{S}$	0.0999	0.0999	0.0998	0.2976	0.4842	0.1000	0.1003
$\overline{D}$	1.2809	1.2576	1.3581	1.7199	2.4911	1.4398	1.1729

BIBLIOGRAPHY

[ABRA70] Abramson, N., "THE ALOHA SYSTEM-Another Alternative for Computer Communications," AFIPS Conference Proceedings, Vol. 37, AFIPS Press, Montvale, N.J., 1970, pp. 281-285.

[ABRA73] Abramson, N.,"Packet Switching with Satellites, " AFIPS Conference Proceedings, Vol. 42, AFIPS Press, Montvale, N.J., 1973, pp. 695-702.

[ABRA75] Abramson, N.,and E.R. Cacciamani, "Satellites: Not Just a Big Cable in the Sky," IEEE Spectrum, Vol. 12, September 1975, pp. 39-40.

[ABRA77] Abramson, N., "The Throughput of Packet Broadcasting Channels," IEEE Transactions on Communications, Vol. 25, January 1977, pp. 117-128.

[AGRA77] Agrawal, A. K., et al., "Analysis of an Ethernet-like Protocol," Proceedings of the Computer Networking Symposium, December 1977. 77CH1252-6C. pp. 104-111.

[ANSI85] ANSI/IEEE, "Carrier Sense Multiple Access with Collision Detection (CSMA/CD), Access Method and Physical Layer Specifications," ANSI/IEEE Standard 802.3, 1985.

[ASOK88] Asok, R., "Networking for Computer-Integrated Manufacturing," IEEE Network, Vol. 2, No. 3, May 1988, pp. 40-47.

[BALA79] Balagangadhar, M. M., and R. L. Pickholtz, "Analysis of a Reservation Multiple Access Technique for Data Transmission via Satellites," IEEE Transactions on Communications, Vol. 27, No. 10, October 1979, pp. 1467-1475.

[BIND75A] Binder, R., et al., "ALOHA Packet Broadcast- A Retrospect," National Computer Conference, AFIPS Conference Proceedings, Volume 44, 1975, pp 201-215.

[BIND75B] Binder, R., "A Dynamic Packet-Switching System for Satellite Broadcast Channels," ICC 75 conference Record, IEEE, New York, 1975, pp. 41-1 to 41-5.

[BOGG88] Boggs, D. R., et al., "Measured Capacity of an Ethernet: Myths and Reality," Proceedings SIGCOM 88, August 1988, pp. 222-234.

[BOGG89] Boggs, D. R., et al., "Errata for "Measured Capacity of an Ethernet: Myths and Reality," " Computer Communications Review, Vol. 19, No. 2, April 1989, p. 10.

[BORG85] Borgonovo, F., L. Fratta, F. Tarini, and P. Zini, "L-Expressnet: The Communication Subnetwork for the C-NET Project," IEEE Transactions on Communications, Vol. COM-33, No. 7, July 1985, pp. 612-619.

[BUX84] Bux, W., "Performance Issues in Local Area Networks," IBM Systems Journal, Vol. 23, 1984, pp. 351-374.

[BUX84] Bux, W., "Performance Computer Communications Systems," Elseveir Science Publishers, N.Y. N.Y.,1984, TK 5105.5.I597, ISBN 0-444-86883-6.

[CAPE79] Capetanakis, J. I., "Generalized TDMA: The Multi-accessing Tree Protocol,"IEEE Transactions on Communication, Vol. 27, No. 10, October 1979, pp. 1476-1484.

[CHLA84] Chlamtac, I., "Exercising the Priority Mechanisms of the Intel 82586 LAN (Ethernet) Component in Support of Applications," Proceedings of the Computer Networking Symposium, 1984, CH2106/84/0000/0147 pp. 147-155.

[CHER82] Cherukuri, R., L. Liang, and L. Louis, "Evaluation of Token Passing Schemes in Local Area Networks," Computer Networking Symposium, December, 78CH1400-1C, 1982, pp. 57-68.

[CHOU78] Chou, W., "Terminal Response Time on Polled Teleprocessing Networks," Proceedings of the Computer Networking Symposium, December 1978. pp. 1-9.

[CHOU85] Choudhury, G. L., and S. S. Rappaport, "Priority Access Schemes Using CSMA-CD," IEEE Transactions on Comunications, Vol. COM-33, No. 7, July 1985, pp. 620-626.

[CLAR78] Clark, D. D., K. T. Pogram, and D. P. Reed, "An Introduction to Local Area Networks," Proceedings of the IEEE, Vol. 66, No. 11, November 1978, pp. 1497-1517.

[COYL83] Coyle, E. J., and B. Lui, "Finite Population CSMA/CD Networks," IEEE Transactions on Communication, Vol. COM-31, November 1983, pp. 1247-1251.

[DEMA76] Demarines, V. A., and L. W. Hill, "The Data Bus in Data Communications," Datamation, Vol. 22, No. 8, August 1976, pp. 89-92.

[DEC80] DEC, Intel, and Xerox Corps., "The Ethernet, A Local Area Network: Data Link Layer and Physical Layer Specifications, Version 1 ed., 1980.

[DHIL87] Dhillon, B. S. , "Reliability in Computer Systems," Ablex Publishing Corporation, Norwood, N.J., 1987, QA 76.9.588 D49, ISBN 0-89391-412-6.

[DONN79] Donnelley J. E., and J. W. Yeh, "Interaction Between Protocol Levels in a Prioritized CSMA Broadcast Network," Computer Networks 3, pp 9-23.

[DOUG83] Douglas, R. H., "A Comparison of IEEE Standard Local Area Networks," Proceedings of the Second Annual Phoenix Conference, March, 1983, pp. 334-335.

[FARO86] Farowich, S. A., "Communicating in the Technical Office," IEEE Spectrum, Vol. 23, No. 4, April 1986, pp. 63-67.

[FAYO77] Fayolle, G., et al., "Stability and Optimal Control of Packet Switching Broadcast Channel," J. ACM, Vol.24, July, 1977, pp. 375-386.

[FINK88] Fink, R., "FDDI as a Backbone for Large Campus Ethernet Networks," Fiber Optics Magazine, March-April, 1988, pp. 25-29.

[FIEL82] Field, J. A., and J. W. Wong, " A Carrier Sense Multipe Access (Collision Detection) System with Global Information," Fall Comcon 82 Proceedings, September, 1982, IEEE Ref. No. CH1796-2/82/0000/0417, pp. 511-520

[FRAN84] William R. Franta and John R. Heath, "Measurement and Analysis of HYPERchannel Networks," IEEE Transactions on Computers, Vol. C-33, March 1984, pp 249-260.

[FERG77] Fergeson, M. J., "An Approximate Analysis of Delay for Fixed and Variable Length Packets in an Unslotted ALOHA Channel," IEEE Tranactions on Communications, Vol. COM-25, July 1977, pp. 644-654.

[FUJI84] Fujiwara, C., et al., "An Analysis of CAMS-CD with Arbitrary Slot Size," GlobeCom 84 Conference Record, 1984, pp. 1166-1170.

[GARD80] Gardner, F. M., and W. C. Lindsey, eds., Special Issue on Synchronization, IEEE Transactions on Communications, Vol. COM-28, No.8, August 1980, pp 1105-1440.

[GOOD88] Goodman, J., A. G. Greenberg, N. Madras, and P. March, "Stability of Binary Exponential Backoff," JACM, Vol. 35, No. 3, July, 1988, pp. 579-602.

[GONS87] Gonsalves, T., "Measured Performance of the Ethernet" in "Advances in Local Area Networks", IEEE Press, New York, 1987, TK 5105 .5 .7.A38, ISBN 0-87942-217-3, pp. 383-410.

[GOPA84] Gopal, P. M.,and J. W. Wong, "Analysis of a Hybrid Token-CSMA/CD Protocol for Bus Networks," Performance of Computer Communications Systems, Elsevier Science Publishers B.V. (North-Holland), 1984.

[GOPE83] Gopen, C., et al., "High Functional VLSI Implementation for CSMA/CD Lans," Proceedings of the Second Annual Phoenix Conference, March, 1983, pp. 319-324.

[GREE87] Greenburg, A. G., P. Flajolet, and R. E. Ladner, "Estimating the Multiplicities of Conflicts to Speed Their Resolution in Multiple Access Channels," JACM, Vol. 34, No. 2, April, 1987, pp. 289-325.

[HAYE78] Hayes, J. F., "An Adaptive Technique for Local Distribution," IEEE Transactions on Communications," Vol. COM-26, No. 8, August 1978, pp. 1178-1186.

[HAYE84] Hayes, J. F., "Modeling and Analysis of Computer Communications Networks," Plenum Press, N.Y. N.Y., 1984, TK 5105 .5 .H39.

[HIRA80] Hirai, K., and Y. Satoh, "Stability of a System with Variable Time Delay," IEEE Transactions on Automatic Control, Vol. AC-25, No. 3, June 1980, pp. 552-554.

[HEAT86] Heatly, S., "A Simplified Discrete Event Simulation Model for an IEEE 802.3 LAN," GlobeCom 86 Conference Record, 1984, pp. 138-142.

[IEEE82] IEEE Project 802 Local Area Network Standards, Draft D, 1982.

[IEEE83A] IEEE, OSI Model, Proceedings of the IEEE, Vol. 71, No. 12, December 1983.

[IEEE83B] IEEE Journal on Selected Topics in Communications, Special Issue on Local Area Networks, ed. K. Kuemmerle, B. W. Struck, and F. A. Tobagi, vol. SAC-4, No. 5, November 1983, pp. 697-960.

[IEEE86] IEEE Journal on Selected Topics in Communications, Special Issue on Network Performance Evaluation, ed. A. Leon-Gracia, vol. SAC-4, No. 5, September 1986, pp. 733-996.

[IEEE87A] IEEE Journal on Selected Topics in Communications, Special Issue on Multiple-Access Networks, ed. V. O. K. Li, vol. SAC-5, No. 6, July 1987, pp. 933-1064.

[IEEE87B] IEEE Journal on Selected Topics in Communications, Special Issue on Interconnection of Local Area Networks, ed. W. Bux, D. Grillo, and N. F. Maxemchuk, vol. SAC-5, No. 9, December 1987, pp. 1377-1520.

[IEEE89] IEEE Communications Magazine, High Speed Network Protocols, eds. Rudin, H. and Williamson, R, vol. 27, No. 6, June 1989, pp. 10-53.

[IKED79] Ikeda, M., and T. Ashida, "Stabilization of Linear System with Time-varying Delay," IEEE Transactions on Automatic Control, Vol. AC-24, No. 2, April 19879, pp. 369-370.

[INTE88] Microcommunications Handbook, Intel Corporation, Santa Clara, CA.,1988.

[ISLE75] Isle, A. P. B., "Stability of Systems with Nonlinear Time-varying Delay," IEEE Transactions on Automatic Control, Vol. AC-20, No. 1, February 1975, pp. 67-75.

[JACO74] Jacobs, I. M., R. Binder, and V. E. Hoversten, " General Purpose Packet Satellite Networks," Proceedings of the IEEE, Vol. 66, No. 11, November 1978, pp 1448-1467.

[JENQ82] Jenq, Y. C., "Theorectical Analysis of Slotted Aloha, CSMA, and CSMA/CD Protocols," GlobeCom 82 Conference Record, Miami, November, 1982, pp. A2.1.1- A2.1.5.

[JURG86] Jurgen, R. K., "Coming from Detroit: Networks on Wheels," Spectrum, Vol. 23, No. 6, June, 1986, pp. 53-59.

[KAHN77] Kahn, R. E., "The Organization of Computer Resources into a Packet Radio Network," IEEE Transactions on Communications, Vol. COM-25, January 1977, pp. 169-178.

[KAHN78] Kahn, R. E., et al., "Advances in Packet Radio Technology," Proceedings of the IEEE, Vol. 66, No. 11, November 1978, pp. 1468-1496.

[KAMI86] Kaminski, M. A., "Protocols for Communicating in the Factory," IEEE Spectrum, Vol. 23, No. 4, April 1986, pp. 56-62.

[KLEI73] Klienrock, L., and S. S. Lam, "Packet Switching in a Slotted Satellite Channel," AFIPS Conference Proceedings, Vol. 42, AFIPS Press, Montvale, N.J., 1973, pp. 703-710.

[KLEI75A] Klienrock, L., and S. S. Lam, "Packet Switching in a Multi-access Broadcast Channel: Performance Evaluation," IEEE Transactions on Communications, Vol. COM-23, April 1975, pp. 410-423.

[KLEI75B] Klienrock, L., and F. A. Tobagi, "Packet Switching in Radio Channels, Part 1: Carrier Sense Multiple Access Modes and Their Throughput-Delay Characteristics," IEEE Transactions on Communications, Vol. COM-23, No. 12, December 1975 pp. 1400-1416.

[KLEI76] Klienrock, L., "Queueing Systems, Volume II: Computer Applications," John Wiley and Sons, Inc., New York, 1976, pp. 360-407.

[KONH74] Konheim, A. G., and B. Meister, "Waiting Lines and Times in a System with Polling," J. ACM, Vol. 21, July 1974, pp. 470-490.

[KRIT86] Kritzinger P. S., " A Performance Model of the OSI Communication Architecture," IEEE Transactions on Communications, Vol. COM-34, No. 6, June 1986, pp 544-563.

[KUMM87] Kummerle, K., "Advances in Local Area Networks," IEEE Press, N.Y. N.Y., 1987, TK 5105 .5 .7.A38, ISBN 0-87942-217-3.

[LAM75A] Lam, S. S., and L. Klienrock, "Packet Switching in a Multi-access Broadcast Channel: Dynamic Control Procedures," IEEE Transactions on Communications, Vol. COM-23, September 1975, pp. 891-904.

[LAM75B] Lam, S. S., and L. Klienrock, "Dynamic Control Schemes for a Packet Switched Multi-access Broadcast Channel," AFIPS Conference Proceedings, Vol. 44, AFIPS Press, Montvale, N.J., 1975, pp. 143-153.

[LAM77] Lam, S. S.,"Delay Analysis of a Time Division Multiple Access (TDMA) Channel," IEEE Transactions on Communications, Vol. COM-25, December 1977, pp. 1489-1494.

[LAM78] Lam, S. S., "A New Measure for Characterizing Data Traffic," IEEE Transactions on Communications, Vol. COM-26, January 1978, pp. 137-140.

[LAM79] Lam, S. S., "Satellite Packet Communication-Multiple Access Protocols and Performance," IEEE Transactions on Communications, Vol. COM-27, No. 10, October 1979, pp. 1456-1466.

[LAM80A] Lam, S. S., "A Carrier Sense Multiple Access Protocol for Local Networks," Computer Networks, February 1980.

[LAM80B] Lam, S. S., "Packet Broadcast Networks-A Performance Analysis of the R-ALOHA Protocol," IEEE Transactions on Computing., Vol. C-29, July 1980, pp. 596-603.

[LITT61] Little, J., "A Proof of the Queueing Formula L=λW," Operations Research., Vol. 9, March-April 1961, pp. 383-387.

[LIU82] Liu, M. T., et al., "Performance Evaluation of Channel Access Protocols for Local Computer Networks," Fall Comcon 82 Proceedings, September, 1982, IEEE Ref. No. CH1796-2/82/0000/0417, pp 417-426.

[MARK78] Mark, J. W., "Global Scheduling Approach to Conflict-Free Multiaccess via a Data Bus," IEEE Transactions on Communications, Vol. COM-26, September 1978, pp. 1342-1352.

[MARK79] Mark, J. W.,and S. F. W. Ng, "A Coding Scheme for Conflict-Free Multiaccess Using Global Scheduling," IEEE Transactions on Communications, Vol. COM-, September 1979, pp. 1353-1360.

[MARA80] Madhav Marathe, "Design Analysis of a Local Area Network," Proceedings of the 1980 Computer Networking Symposium, IEEE, 1980. PP. 67-81.

[MARK80] Mark, J. W., "Distributed Scheduling Conflict-Free Multiple Access for Local Area Communications Networks," IEEE Transactions on Communications, Vol. COM-28, December 1980, pp. 1968-1976.

[MAGU88] Maguffin, L. J., L. O. Reid, and S. R. Sparks, "MAP/TOP in CIM Distributed Computing," IEEE Network, Vol. 2, No. 3, May 1988, pp. 23-31.

[MEDI83] Meditch, J. S., and C. T. A. Lea, "Stability and Optimization of the CSMA and CSMA/CD Channels," IEEE Transactions on Communications, Vol. COM-31, November 1983, pp. 763-774.

[MESS86] Messenger, G., and Ash, M., "The Effects of Radiation on Electronic Systems," Van Nostrand Reinhold Company, N.Y. N.Y., 1987, TK 7870.M4425, ISBN 0-442-25417-2.

[METC76] Metcalf, R. M., and D. R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," Communications of the ACM, Vol. 19, No. 7, July 1976, pp. 395-404.

[MEYE86] John W. Meyer, "SAE AE-9B Draft Standard High Speed Token Passing Data Bus for Avionics Applications," Proceedings of the 7th AIAA/IEEE Digital Avionics Systems Conference, IEEE, 1986, pp. 234-241

[MILS87] MIL-STD-1553 Designer's Guide, ILC Data Device Corporation, Bohemia, N.Y.,1987.

[MITT85] Mittal, K., and P. O'Reilly, "Comparative Evaluation of Different Access Schemes for Local Ring Networks," Tenth Conference on Local Computer Networks, October, 1985, pp. 114-122.

[MOSH88] Mosher, Timothy, Glenn McNeil, and Michael Nehl, "DISTRIBUTED DATA BUS STUDY FOR SPACE TRANSPORTATION", Procurement Instrument Identification Number F33601-85-D0070, Report number AFWAL-TR-87-3111.

[MURR84] Murry, H. F., "Communications Network Modeling Figure of Merit," GlobeCom 84 Conference Record, 1984, pp. 831-835.

[NG77] Ng, S. F. W., and J. W. Mark, "A Multiaccess Model for Packet Switching with a Satellite Having some Processing Capability," IEEE Transactions on Communications, Vol. COM-25, January 1977, pp. 128-135.

[NUTT82] Nutt, G. J., and D. L. Bayer, "Performance of CSMA/CD Networks Under Combined Voice and Data Loads," IEEE Transactions on Communications, Vol. COM-30, No.1, January 1982, pp. 6-11.

[OKAD84] Okad, H. Y., Normura, Y., Nakanishi, Y., "Multi-channel CSMA/CD Methods in Broadband-Bus Local Area Networks," GlobeCom 84 Conference Record, 1984, pp. 642-647.

[PHIL87] Phillips, B. W., "Manage MIL-STD-1553B Bus with just one Chip," Electronic Design, Vol. 35, No. 24, October 15, 1987, pp. 51-54.

[PICK86] Pickholtz, R. ed., "Local Area and Multiple Access Networks," Computer Science Press, Rockville Maryland, 1986, TK 5105 .7 .L58, ISBN 0-88175-143-X.

[PLEI88] Pleinevaux, P., and J. D. Decotignie, "Time Critical Communication Networks: Field Busses," IEEE Network, Vol. 2, No. 3, May 1988, pp. 55-63.

[PRIT77] Pritchard, W. L., "Satellite Communication- An Overview of the Problems and Programs," Proceedings of the IEEE, Vol. 65, March 1977, pp. 294-307.

[PROT84] "Operations and Maintenance Manual for the ProNet Model p1200 Multibus Local Area Network System," Proteon Inc., 1984.

[PROT86] "ProNet-10 Series p320x Counter-Rotating Ring Operation and Maintenance Manual," Proteon Inc., 1986.

[PUEN71] Puente, J. G., W. G. Schmidt, and A. M. Werth, "Multiple-Access Techniques for Commercial Satellites," Proceedings of IEEE, Vol. 59, February 1971, pp. 218-229.

[RELI83] M.J. Relis, "New MIL Standard Opens Way for Optical Links in Aircraft," Electronic Design, Vol. 31, No. 16, August 4, 1983, pp. 153-156.

[ROBE73] Roberts, L.,"Dynamic Allocation of Satellite Capacity Through Packet Reservation," AFIPS Conference Proceedings, Vol. 42, AFIPS Press, Montvale, N.J., 1973, pp. 711-716.

[ROBE78] Roberts, L.,"The Evolution of Packet Switching, "Proceedings of IEEE, Vol. 66, November 1978, pp. 1307-1313.

[SAUE82] Sauer, A. M., "A Local Area Multiprocessor Network," IEEE 1982 pp. 187 CH1739-2/82/0000-0187

[SAUE78] Sauer, C. H.,, "Passive Queue Models of Computer Networks," Proceedings of the Computer Networking Symposium, December 1978, pp. 11-15.

[SEGA83] Segal, A., "Distributed Network Protocols," IEEE Transactions on Information Theory, Vol. IT-29, January 1983, pp. 23-55.

[SHOC80] Shoch, J. F., and J.A. Hupp, "Measured Performance of an Ethernet Local Area Network," Communications of the ACM, Vol. 23, No. 12, December 1980, pp. 711-721.

[SIGN88] Signorile, R. P., J. LaTorette, and M. Fleisch, "MBRAM - A Priority Protocol for PC Based Local Area Networks," IEEE Network, Vol. 24, No. 4, July, 1988, pp. 55-59.

[SOHR84] Sohraby, K., Mulle, M., Venetsanopoulos, A., "Why Analytical Models of Ethernet -Like Protocols are so Pessimistic," GlobeCom 84 Conference Record, 1984, pp. 659--664.

[SPIE86] James E. Spieth and Walter D. Seward, "Simulation Model of a High-Speed Token-Passing Bus for Avionics Applications," Proceedings of the 7th AIAA/IEEE Digital Avionics Systems Conference, IEEE, 1986.

[SPIT86] Spitzer, C.R., "All-Digital Jets are Taking Off," Spectrum, Vol. 23, No.9, September, 1986, pp. 51-56.

[STALL84] William Stallings, Local Networks: an Introduction, MacMillan Publishing Co., 1984.

[STALL85] William Stallings, Data and Computer Communications, MacMillan Publishing Co., N.Y. N.Y., 1985.

[STALL87] William Stallings, Handbook of Computer Communications Standards Volume 2: Local Network Standards, MacMillan Publishing Co., N.Y. N.Y., 1987, TK 5105 .S732.

[SZUR78] Szurkowski, E., "A High Bandwidth Local Computer Network," Proceedings of Fall COMPCON 78, Washington D.C., September, 1978, pp. 98-103.

[TAKA85] Takagi, H., and L. Kleinrock, "Throughput Analysis for Persistent CSMA Systems," IEEE Transactions on Communications, Vol. COM-33, No. 7, July 1985, pp. 627-638.

[THOR83] Thornton, J. E., and G. S. Christensen, "Hyperchannel Network Links," IEEE Computer, Vol 16, No.9, September 1983, pp. 50-54.

[TOBA 75] Tobagi, F. A., and L. Klienrock, "Packet Switching in Radio Channels: Part II. The Hidden Terminal Problem in Carrier Sense Multiple Access and the Busy Tone Solution," IEEE Transactions on Communications, Vol. COM-23, No. 8, December 1975, pp. 1417-1433.

[TOBA 76] Tobagi, F. A., and L. Klienrock, "Packet Switching in Radio Channels: Part III-Polling and (Dynamic) Split Channel Reservation Multiple Access," IEEE Transactions on Communications, Vol. COM-24, No. 8, August 1976, pp. 832-845.

[TOBA 77] Tobagi, F. A., and L. Klienrock, "Packet Switching in Radio Channels: Part IV-Stability Considerations and Dynamic Control in Carrier Sense Multiple Access," IEEE Transactions on Communications, Vol. COM-24, No. 8, October 1977, pp. 1103-1119.

[TOBA80] Tobagi, F. A., "Multi-access Protocols in Packet Communications Systems," IEEE Transactions on Communications, Vol. COM-28, No. 4, April 1980, pp. 468-488.

[TOBA87] Tobagi, F. A., "Performance Analysis of Carrier Sense Multiple Access with Collision Detection" in "Advances in Local Area Networks", IEEE Press, New York, 1987, TK 5105 .5 .7.A38, ISBN 0-87942-217-3, pp. 318-339.

[TROP81] Tropper, C., " Local Computer Network Technology," Academic Press, San Francisco CA., 1981.

[TURN85] Turner, W., "Protocol Implementation Strategies in Local Area Network Access Units," AFIPS Proceedings, Vol. 54, 1985, pp. 611-617.

[TURN87] Turner, D. B., R. D. Burns, and H. Hecht, "Designing Micro-Based Systems for Fail-Safe Travel," Spectrum, Vol. 24, No. 2, February, 1987, pp. 58-63.

[VOEL86] Voelcker, J., "Helping Computers Communicate," IEEE Spectrum, Vol. 23, No. 3, March 1986, pp. 61-70.

[WEIS78] Weissler, R., et al., "Synchronization and Multiple Access Protocols in the Initial Satellite IMP," Proceedings of Fall COMPCON 78, Washington D.C., September, 1978, pp. 356-362.

[WEST72] West, L. P., "Loop-Transmission Control Structures," IEEE Transactions on Communications, Vol. COM-20, No.3, Part II, June 1972, pp. 531-539.

[WOOD79] Wood, D. C., S. F. Holmgren, A. P. Skelton, "A Cable-Bus Protocol Architecture," Sixth Data Communications Symposium, November, 1979, pp. 137-146.